The PN Table Manager

10th Semester IMM Project

Julien Boriachon and Charles Dagouat

June 4, 2008

The PN Table Manager
by Julien Boriachon and Charles Dagouat

Published June 4th, 2008
Copyright © 2008 The PN Table Manager Team (08gp1071)

AALBORG UNIVERSITY
DEPARTMENT OF ELECTRONIC SYSTEMS

Niels Jernes Vej 12 ▪ DK-9220 Aalborg East Phone 96 35 86 35

TITLE:
The PN Table Manager

THEME:
Integration of Information in Multi-Modal Systems

PROJECT PERIOD:
February 2008 to June 2008

PROJECT SEMESTER:
10th Semester

PROJECT GROUP:
1071

GROUP MEMBERS:

Julien Boriachon

Charles Dagouat

SUPERVISORS:
Alexandre Fleury
Lars Bo Larsen
Rasmus Løvenstein Olsen

NUMBER OF COPIES: 4

NUMBER OF PAGES: ??

CONCLUDED: 04-06-2008

SYNOPSIS:

Since several years, firms are working towards improve the usability parts of the services/products they are delivering to end-users. Examples of this evolution are for instance the iPhone, or the Microsoft's project Surface. Both of them come along with a new interaction paradigm letting the user take advantage of the embedded touch screen. On the other hand, electronic devices come more and more with short range wireless modules embedded. Thus, these devices are able to organize themselves/ be organized into networks called Personal Area Networks (PAN) in order to offer to the user a set of services. This kind of network is characterized by the small geographic area covered by the network. These two main evolution will bring some new challenges, both commercial (which are taken up by companies) and also research ones, to fully investigate the potential offered by Personal Networks.

The main problem we are trying to address is the lack of (networking) homogeneity regarding personal devices which belongs to one user. For instance, from the PAN concept, we are moving to a Personal Network (PN) one, being composed of several PAN located in different areas. The Magnet Beyond project is one of these projects addressing this challenge. The technology developed by Magnet is quite evolved, but it still misses a good way to manage devices' services and their membership to the PN in a way that is intuitive to understand for everyday users. Our main goal is thus to investigate how this can be achieved through the development of a system, which enables intuitive graphical representation and manipulation of PNs representation. Our project will take advantage of a 2007 master thesis project realized by Alexandre Fleury wherein a large-size multitouch screen was embedded in a table top.

The result will be to bring to the end-users a multi-touch screen table allowing PN users to manage in a friendly way services and resources shared. Our software will take advantage of the multitouch screen to allow users to use their fingers to go through the different functionality offered in our application. Basically, it will let the user (un)share services between devices belonging to the PN.

key words: Master Thesis, tabletop display, deme-Touch, Aduna, Clustermap, cluster map, multi-touch.

Contents

List of Figures

List of Tables

List of Examples

Abstract

Motivation: Since several years, firms are working towards improving the usability parts of the services/products they are delivering to end-users. Examples of this evolution are for instance the iPhone™, or the Microsoft's project Surface™. Both of them come along with a new interaction paradigm letting the user take advantage of the embedded touch screen. On the other hand, electronic devices come more and more with short range wireless modules embedded. Thus, these devices are able to organize themselves/ be organized into networks called Personal Area Networks (PAN) in order to offer the user a set of services. This kind of network is characterized by the small geographic area covered by the network. These two main evolutions will bring some new challenges, both commercial ones (which are taken up by companies) and research ones, to fully investigate the potential offered by Personal Networks.

Problem statement: The main problem we are trying to address is the lack of (networking) homogeneity regarding personal devices which belong to one user. For instance, from the PAN concept, we are moving to a Personal Network (PN) one, being composed of several PAN located in different areas. The Magnet Beyond EU project is one of these projects addressing this challenge. The technology developed by Magnet is quite evolved, but it still misses a good way to manage devices' services and their membership to the PN in a way that is both intuitive and understandable for the everyday users. Our main goal is thus to investigate how this can be achieved through the development of a system, which enables intuitive graphical representation and manipulation of PNs representation. Our project will take advantage of a master thesis project [Fleury - 2007] realized by Alexandre Fleury wherein a large-size multi-touch screen was embedded in a table top.

Approach: Our approach will be to uncover the Magnet project through the reading of a huge amount of documentation already written. We will also present the work done around the multi-touch interface, and the human computer interaction made possible through the use of such a screen. We will describe the choices made along this period in the design and implementation parts, and finally, present the results of the tests done.

Results: As a result, we want to bring to the end-users a multi-touch screen table allowing PN users to manage in a friendly way services and resources shared. Our software will take advantage of the multi-touch screen to allow users to use their fingers to go through the different functionality offered in our application. Basically, it will let the user (un)share services between devices belonging to the PN.

Conclusions: Unfortunately, we are not able to take advantage of the multi-touch capability of the tabletop display, because of the Aduna API we are using to display devices and their services. Even though, we are able to display devices and services in a user-friendly way. This user interface is not much evolved but we tried to simplify it in order to make the end-users' life easier. Thanks to this UI, it is quite simple to understand relationship between devices at a PN service level.

Dedication

We would like to thanks all the people who helped us during this semester, and particularly:

- Our supervisors Lars Bo Larsen, Rasmus Løvenstein Olsen and Alexandre Fleury who spent some time to advice us along the semester, and who gave us precious inputs
- Our IMM secretary Charlotte Skindbjerg Pedersen, who did everything possible in order to help us during our day-to-day life at the university
- All the French students (Yoni Bauduin, Christophe Biglete, Delphine Blondeau, Alexandre Gimenez, Jimmy Mattiuzzo and Fadi Zaran) we worked with all along this semester
- And finally, all the other people from the university who helped us when we were experiencing difficulties.

Preface

Purpose of the report: This report is part of the 10th semester project in Intelligent Multimedia at Ålborg University. It has been written by the project group IMM 08gr1071 during the spring semester of 2008. This one is the continuation of the 9th semester whose theme was: *Integration of Information in Multi-Modal Systems*. Its purpose is to give students an insight into working with advanced *intellimedia* systems.

This report contains the overall description of the project and the choices made to realize it. The intended audience is not expected to have any prior knowledge about Human-Computer Interaction, but a general development knowledge is still recommended.

Report's organization: The structure layout of this document is quite simple. In a brief introduction, we present our project (definition, main issues). Then in a first chapter, we present the background of this project: the reason why we decided to work on such a project, the delimitation imposed and some scenarios illustrating the basic use cases of our product. Then the report is divided into two parts: the first one about the theoretical analysis of this subject, and the second one about the practical approach we had during its realization. Each of these parts are sub-divided in several chapters corresponding to the steps we followed during this semester (Pre-analysis and Analysis for the first part, Design, Implementation and Tests for the second one). Finally, the last chapter about our conclusion and the perspective of this project. After these different parts, you can find after these different parts the appendixes: mainly a glossary, a bibliography and an index.

Features and resources: Throughout the report, the references are reported as following: [author - year of publication]. The bibliography is available at the end of this document, in the appendix, under the headline Bibliography.

Figures and tables are marked by primary and secondary indexes. The primary index relates to the chapter number or the appendix letter, and the secondary one is a consecutive number for each chapter or appendix. For instance, Figure 1.3 refers to the third figure in the first chapter.

Figures and program-listing can contain call-out. These ones are represented by black numbered dots. Explanations related to these call-out stand just under the figure. They are also introduced by numbered dots.

In the *PDF* version of this document, words in red are linked to their definition in the glossary at the end of the report. The ones in blue are either links to a website or an online resource, or if they are surrounded by "[]", they are links to references in the bibliography.

This reports will come with a CD-Rom containing the binaries as well as the source code of the application we have developed and its documentation (JavaDoc). It will also come with some PDF version of the documents we have read along this semester. Finally, both this report and the source code will be available on the Project Library website [1] .

Technology notice: This present document has been written using a Docbook scheme and is published using the Dblatex/LaTeX tools and a bunch of XSL stylesheet.

The images in this document are free of rights and have been retouched with the Gimp. The graphics have been made with Dia 0.96.

[1] http://projekter.aau.dk/projekter/

Introduction

During the first semester in the Aalborg University, we have been introduced to the notion of *intellimedia*. All a long this semester, we have followed lectures regarding interactions between the user and the system. These ones were really interesting, and they inspired us when we had to chose the subject of our master thesis project.

Problem statement: During the 60s, quite a long time ago related to the informatics and networks history, researchers invented the computer network as we know it today[2]. Then they invented the Local Area Network (LAN) denomination to present to their peers a restricted network composed by all the computers connected in the office or in the house. This kind of network was not as wide as was the Internet, which at the opposite was denominated by Wide Area Network (WAN).

Up to now, with the appearance of the Wireless technologies such as the Bluetooth, or more recently the ZigBee, appears the Personal Area Networks (PAN). Consequently, most of the new devices are connected to one or more networks. Thanks to this, these networking devices can offer extended services empowered by their communication facility. But as these devices are part of LANs (for instance, a computer, a printer or even a fridge), or even PANs (for instance a cell phone or bluetooth headphones), services they are providing can be accessible only when you are in the same proximity area. Thus, one of the major challenge for the researchers is to invent a new networking model letting the user take advantage of every services published by their devices, even if these one are not in the current PAN. Magnet Beyond is one of these research projects who want to redefine the networks and its usage by describing a user centric network. This one offers services which are available to the user wherever and whenever he is. This network model which is called the Personal Network (PN) extends the PAN model. It is made up of several PAN called clusters which can communicate with others.

The main problem we are trying to address is the lack of user friendly management tool. Effectively, companies and universities working on this project are mainly focusing on both the technical and scientific part of the project (i.e. improvement of the device antenna, in saving battery lifetime or even communication security) and on defining use cases of such a new technology. This project has currently one program called *pnmanager* to allow users to manage their devices interaction within the PN. But from our point of view, this one is not really user friendly. Consequently, we have decided to fulfill this lack by taking advantage of the new orientation given by Apple and Microsoft with their respective projects: the iPhone™ and the Surface™ table. Both of them rely on touch screens enabling users to interact with the device in a fluent way, using only fingers as pointers.

Motivation: The main topics which motivated both of us for working on such a subject are studying user interactions with touch screen related devices, and the Magnet Beyond project. These two domains are really interesting because they are related to news technologies, their usages, and the design of systems using such technologies.

The Magnet project in itself is interesting because of several points, such as the domains implied within the project. These ones are going from the work on the physical part of the devices (mainly aiming to improve performance), to the definition if the technologies upon which rely the network. It also define the use cases, the conceptual model and the scenarios. We think that working on such a project is really interesting because it will probably shape the evolution of the mobile communication usage.

On the other hand, working on designing an application using a touch screen is really a good opportunity because it implies taking into account quite a lot of things we usually do not care about when we design a classical interface.

Also, working on such a project is interesting because, as students and developers, it will let us discover and deal with subjects that are not yet widely available. The market is still to be conquered and that is why it is interesting to develop a product for a market that does not really exist yet and is likely to start growing in the near future.

Main objectives: The main goal we are trying to reach is to realize an application built upon Alexan-

[2] ARPANET was launched in 1969.

dre Fleury's multi-touch table top display. This one is the result of the master thesis project. Our software will be part of a PN network, and will be able to manage services published by devices.

1 Project Presentation

This chapter presents the basic ideas we thought about when we chose our master thesis project. This one requires the use of the table project lead in 2007 by Alexandre Fleury. Our main idea is to use the multi-touch display table as a mean to present graphically to user(s) information about their PN device(s), and their associated capability.

To present our ideas to the reader as they were at the beginning of the semester, we will firstly describe the purpose of our project. Secondly, we will explain why we though about working on such a subject. Then, before introducing the delimitation of the project, we will describe the scenarios. And finally, we will close this chapter by writing down the main issues we plan to face along this project.

1.1 Purpose of the project

The purpose of this semester is to write our master thesis and the related project. The goal we want to reach is to add to the Magnet project a new tool to manage a user's Personal Network in a more friendly way than with the current *pnmanager* tool. To reach this goal, we are going to take advantage of the tabletop display realized by Alexandre Fleury [Fleury - 2007] in his master thesis project. This table has two functions: firstly a screen function to display things on top of it, and secondly a pointer function. This one is made to detect fingers letting the user interact with the computer using this setup. Thus, the main goal we want to reach during this semester is to familiarize enough with the Magnet project and with the multi-touch tabletop display in order do be able to run our project on the top of this architecture.

Our system will let the user interact with his PN in a friendly way because we plan to design it taking into account several criteria such as:

- **Intuitive and user friendly** interface, using drag & drop for example
- **Icons metaphors** to represent PN devices, such as computers, mobile phones, printers, . . .

This application will let the user mainly manage services published by the PN devices. For instance, a computer can share with other devices files, a camera can share with other devices either the capability to take pictures, or stream movies. But in a Personal Network, these services can be share or not at a device level. This means that a smartphone can share its address book with one special computer, but not with another one. Thus our software will let view and edit the current setup of his PN.

1.2 Why did we chose this project

PN devices are becoming more and more usual. Actually, there is more and more terminals empowered by networking capabilities. These ones are no more restricted to the professional market, neither to a particular type of devices. There is more and more devices which can go on the network using the new technologies such as Bluetooth©, WiFi, or even Zigbee.

We have chosen to do such a project to cope with the interests of both of members of the team. As this project involves both a network part, and a Human-Computer interface one, it covers a lot of field of activities going from the user interaction, to the security part of the network communications. Moreover, this project is related to the future use of the communication terminals, going from mobile phones to computers devices. The idea of working on such a new technology is really motivating.

1.3 Basic scenario

These basic scenarios are made in order to gave us directions to follow. They are also useful to define the main functions of the prototype we want to realize. These guidelines are going to evolve along the project. The steps through which we pass correspond to the main chapters of this report (Chapter 2 - Pre-Analysis and Chapter 3 - Analysis)

The basic scenario for our project will be:

- The user comes around our table with his mobile device belonging to his PN.
- The table detects the device presence, and displays on the table the device profile. This can be done either by drawing a picture of the device, or by a simple schema/text.
- The next step will depend on the user's goal:
  - he will basically be able to configure his PN by moving devices on the screen to create links between them. For instance, he will be able to manage data access such in order to share music/video/documents files between devices,
  - another possibility will be to allow the user to configure his devices using a virtual keyboard,
  - last, but not least, the user might be able to create or modify files, by providing him some kind of writer/word software.
- When the device leaves the table detection area, the device should disappear from the table screen.

Figure 1.1 Basic description of our system

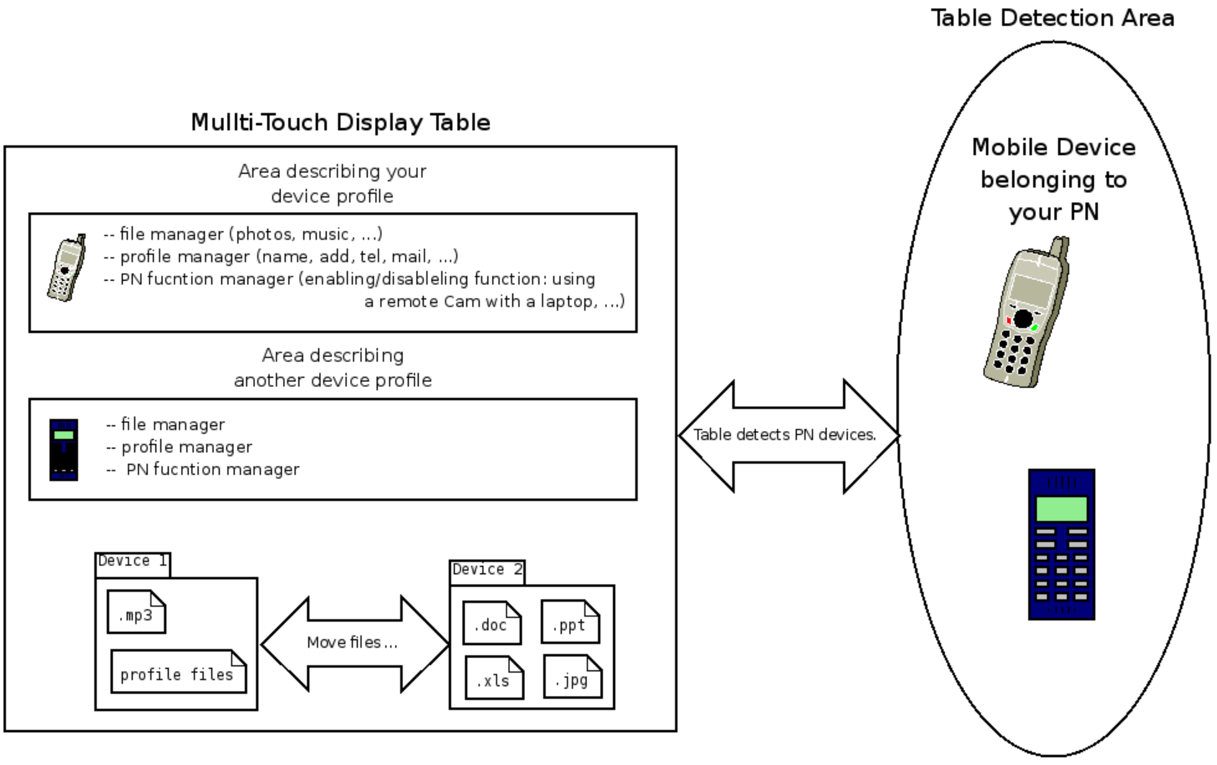


Table Detection Area
Mullti-Touch Display Table
Area describing your device profile
-- file manager (photos, music, ...)
-- profile manager (name, add, tel, mail, ...)
-- PN fucntion manager (enabling/disableling function: using a remote Cam with a laptop, ...)
Area describing another device profile
-- file manager
-- profile manager
-- PN fucntion manager
Device 1
.mp3
profile files
Move files ...
Device 2
.doc
.ppt
.xls
.jpg
Table detects PN devices.
Mobile Device belonging to your PN

First thought of our system.

1.4 Delimitation of the project

Regarding that at the time we are writing this report, the Magnet project is in constant evolution, and that it is can be quite hard to build a product in its final state with some of its required component not finalized, we are going to realize a project which has only a demonstration purpose. The main goal we want to reach is to build a piece of software on top of the architecture defined by Alexandre Fleury in [Fleury - 2007] for his tabletop system, in order to allow end-users to manage their PN devices.

Thus, the main limitation we are going to face will be the Magnet Project. We can not modify any piece of code of this project. In fact, this one is only a tool in our project. Moreover, we do not know the project as much as needed to try to adapt it to our project. Secondly, because modifying any small part of the source code can have repercussions on all the other part of the project…

Of course, another limitation will the limited time we have to realize such a project.

1.5 Main issues regarding the feasibility of our project

This project can be split into several elements which are going to interact with each others. The main ones are:

- **The physical tabletop display:** We will mainly need to configure it to fit with our requirements. Also, we will maybe need to improve it regarding some issues with the finger detection accuracy or speed of the engine.
- **Our software component:** Relying on this first element, our program will need to get the fingers coordinates from the table and will be able to send to the table the screen to display. This software will also rely on the Magnet project to be able to be inserted into a Personal Network, and to communicate with other component of this network.

These issues will be discussed in details in Chapter 2 - Pre-Analysis and Chapter 3 - Analysis, where solutions will be proposed.

1.6 Problem statement

Regarding all the previous sub-sections, we can formulate the question we will try to solve in this project.

- What would be the best user-friendly way to represent devices and services within a Personal Network?
- Is it possible to improve the already existing *Magnet command line interface*?
- How can we benefit from the use of a multi-touch table?

We can reformulate these questions into only one which could be "*Is it possible to use a tabletop display to create an intuitive interface for a PN management system?*".

Part I

Analysis Part

2 Pre-Analysis

The main purpose of this chapter is to give an overview of the existing tools and technologies that are going to be used in our project, or to influence it. The following information should give the basis to understand the way our system's architecture is going to be designed.

This project will be mainly composed of two components:

- **The table module** which is going to provide two services: detect fingers, and display information
- **Our software module** which is going to gain advantage of the previous module, to display to the user the provided information related to his PN. Thus this module's main role is to acquire information about the PN and from the table finger detector and to analyze these information in order to behave accordingly to the user actions on the table.

These two modules are not built from scratch. We currently rely on the work already done in several domains, such as the Magnet Beyond project for instance. Each of these domains will be discussed in the following sub-parts.

2.1 Magnet Beyond

2.1.1 Project presentation

Regarding the organization of the project, this section refers mainly to [Magnet - 2006] and to [WP1 - 2006]. Regarding the PN use, this section refers to [Schultz et al - 2006]. The following written content is a rephrasing in our own words of the various information we found within these documents. Finally, if the reader want to get a quick overview of the Magnet Beyond project, we recommend the read of [Magnet - 2008].

Magnet Beyond [1] is a worldwide project and a continuation of the Magnet Project. It's a R&D project within Mobile and Wireless Systems which involves lots of partners (more than 30) from Universities and Research centers to SMEs and a total budget of 35M€ in a 4 year period. This project is aiming at re-defining the use of networking devices in the everyday life. To do this, the project is divided into several work-groups. Each of them is assigned a different task going from the optimization of the Air-Interfaces for the PAN networking, to the definition of the PN platforms and the PN networking, to the Security and privacy of communication into the PN, to finally the specifications, design and implementation of pilot services.

The project comes from a simple remark. More and more users own devices empowered by their embedded networking module. But all these devices are not always in the same place where the user is. Nowadays, a lot of users have a computer at home, but also in their office. They also own a mobile phone

[1] www.ist-magnet.org

or a smartphone which is able to communicate with both of these computers over the Bluetooth link. Moreover, these devices do not offer exactly the same services. For instance, the office computer can be able to print documents, whereas the home computer which is not physically linked to any printers can not offer this service. Consequently, services provided by these devices are not always usable regarding the place where you are.

Thus regarding this statement, the goal of the project is to define a new kind of networking architecture in a user-centric way. That is to say that the network, which is made of all the devices owned by a user, must provide services wherever and whenever the user is. To reach this goal, the project is designed in a multi-network, multi-device, and multi-user way. This architecture enable the user to securely communicate over the network(s), and to gain access to services provided by either his PN or other networks.

2.1.2 Personal Network Organization

This section refers mainly to [Sivarajah et al - 2005] and to [Petrova et al - 2005]. The following content written is a rephrasing in our own words of the various information we found within these documents. To simply get a glimpse of this concept, we recommend the read of [Hoebeke et al], [WP2 - 2006] and finally [WP6 - 2006].

The organization of the PN can be quite complicated regarding the interactions needed to design a user-centric network. Basically, the network is made of devices and nodes. Devices are all the communicating entities. That is to say that any device embedding e.g. a Bluetooth module can be considered as a device in a Personal Network. On the other hand, nodes are devices implementing either (or possibly both) IPv4 or IPv6 protocols. Both the devices and nodes are considered Personal if a trust attribute has already been established. Usually, they are owned by a MAGNET user.

All these personal devices and nodes are connected each other in a Local or even in a Personal Area Network (LAN / PAN). These small networks are called Clusters in the MAGNET terminology. As reminder, one of the goal to reach in the MAGNET project is to create a user-centric network. That is to say that the goal is to let the user use all his devices in a secure way within a virtual and private network (VPN), over the physical interconnecting structures [2] . This implies that nodes located into different clusters must be able to communicate and exchange services. To do this, MAGNET introduces several special node status:

- **The Gateway Node** (GN) is a Personal Node within a Cluster that enables connectivity to other nodes and devices outside the Cluster.

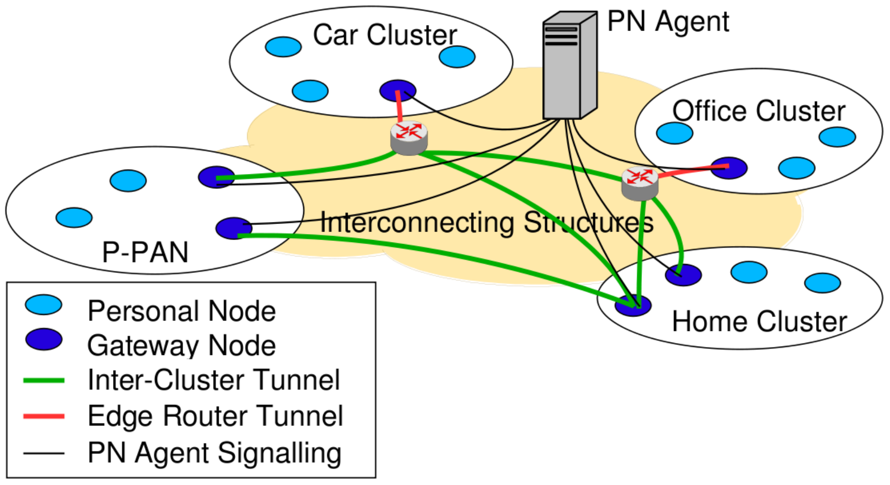
- **The Edge Router** (ER) which is not compulsory, can aid the PN organization and networking, by outsourcing some of the PN functionality to the edge router located in the interconnecting structures. This way, the gateway node will delegate the PN routing and networking to the edge router.

- **The PN Agent** is used to allow different clusters of the same PN to interconnect each others either directly via the gateways, or if a Edge router is used, by the ER.

The internal cluster organization is made automatically, relying on the physical network. The organization of the clusters at the PN level is either managed by the Gateway Node, or by the Edge Router, depending on the availability of the ER.

[2] Public, private or shared wired, wireless or hybrid networks such as a UMTS network, the Internet, an intranet or an ad-hoc network.

Figure 2.1 Personal Network organization



[Sivarajah et al - 2005] (p13) - Organization of the PN.

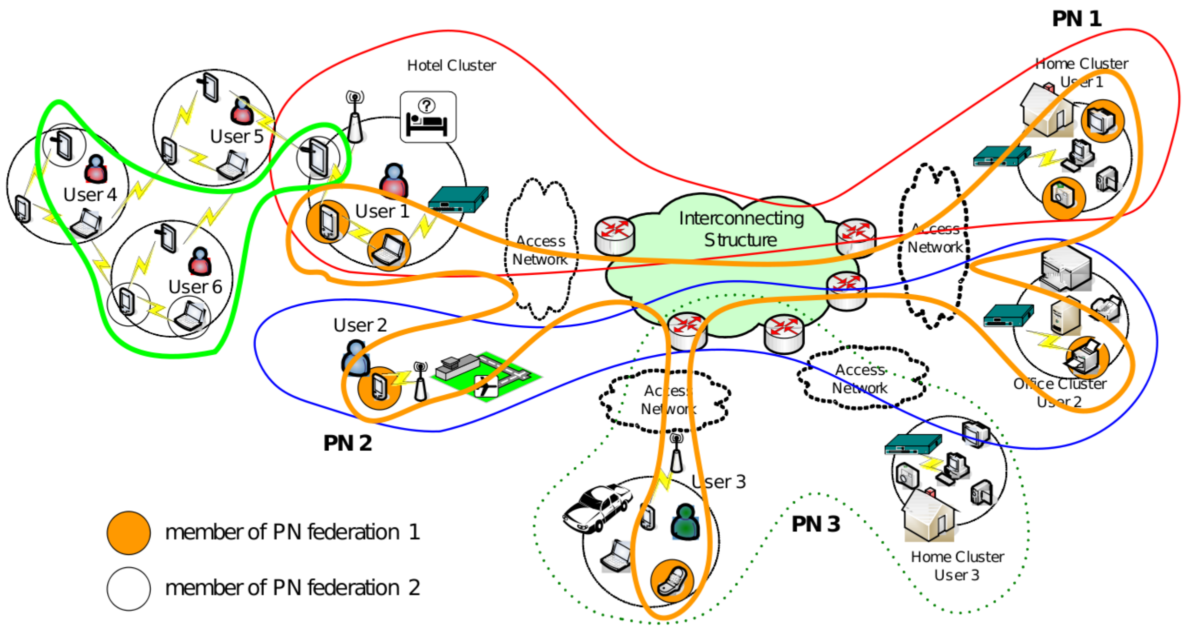
2.1.3 Personal Network Federation

This section refers mainly to [Sivarajah et al - 2005] and to [Ghader et al - a]. The following content written is a rephrasing in our own words of the various information we found within these documents. To simply get a glimpse of this concept, we recommend the read of [WP2 - 2006].

A PN Federation or PN-F, is a federation of several Personal Networks. It enables secure cooperation between different PNs, making selected service(s) and resource(s) available to selected receiver(s). A PN-F can be set up between a user´s PN and a public PN allowing this user to benefit from services offered by the public PN. For instance, our table could be set-up to offer a public service allowing users to manage interactions between their PN nodes and devices.

Finally, the organization of the PN at the federation level is managed by the same entities as at the PN level, by the GN or by the ER, depending on its use.

Figure 2.2 Personal Network Federation



[Sivarajah et al - 2005] (p84) - PN Federation.

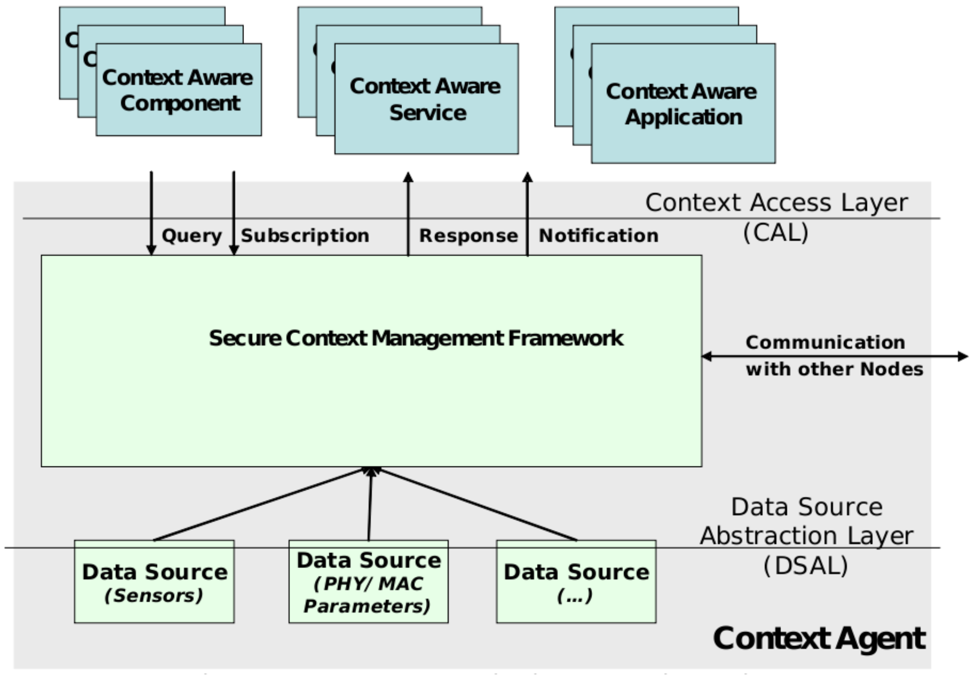
2.1.4 Context & Services

This section refers mainly to [Sivarajah et al - 2005] and to [Ghader et al - 2004]. It also refers to various articles: [Balken et al - 2006], [Bauer et al] and [Ghader et al]. The following content written is a rephrasing in our own words of the various information we found within these documents. To simply get a glimpse of this concept, we recommend the read of [WP2 - 2006].

Being aware of the context allows the system to be more sensitive, adaptive and responsive to both its needs and the user ones. Thus, in the MAGNET project, information characterizing either a situation, a status, a person, an object or even a service is part of a *context*. User, environment and network context are used to make the research easier and the use of services provided inside and outside of the PN. This information is managed in the **Secure Context Management Framework** (SCMF) and more precisely by the **Context Management Node** (CMN). This task is usually assigned to the gateway node, however the selected node is elected among all the nodes regarding its score which is computed using node's metrics (computational devices, battery, bandwidth, ...).

Nowadays, there are potentially more and more data sources for this CMF, because all the objects in our everyday life come with communication modules embedded. For instance, the home automation is a great example of the current progress in this field. With the appearance of more and more data sources for this framework, it will become more and more usefull (as well as reliable regarding the global evolution of the framework).

Figure 2.3 Context Management Framework


Context Aware Component
Context Aware Service
Context Aware Application
Context Access Layer (CAL)
Query
Subscription
Response
Notification
Secure Context Management Framework
Communication with other Nodes
Data Source Abstraction Layer (DSAL)
Data Source (Sensors)
Data Source (PHY/ MAC Parameters)
Data Source (...)
Context Agent

[Sivarajah et al - 2005] (p64) - Context Management Framework.

Context is used to move from a network-centric design to a user-centric one. This design allows the system to react to the environment. For instance, if a user approaches a movie theater, his smartphone could propose him to easily access a service allowing him to view the list of movies currently played.

There are different kind of services. Firstly, there are the personal ones, which are provided by either personal nodes or devices. For instance, a computer can provide a service to print documents. This service will be personal because you do not want to let just anybody print just anything. Thus, this kind of services is only provided within your private PN. Personal services are private and need the establishment of a trust relationship. At the opposite, there are the public services which can be reach without requiring the establishment of this trust relationship.

All these services are managed by the **Service Management Node** (SMN) which is a selected node responsible for a centralized service discovery both on the local and remote service discovery components, within the PN or not.

2.1.5 Magnet technologies

Magnet is not built from scratch. It relies on a lot of existing technologies. TCP/UDP and IPv4/6 layers are the basis of both nodes and interconnecting structures communications. Bluetooth is the main wireless communication technology for the devices networks, whereas WiFi is the main one for the nodes networks. Magnet also relies on technologies such as SSL, VPN, SHA1/2 to ensure privacy and security. Finally, it relies on upper layer protocols such as HTTP/FTP, XML-RPC/SOAP for communication between entities, UPnP for service discovery and XML, RDF and WSDL for description purpose.

2.1.6 Magnet scenarios: the Magnet.Care example

This section refers mainly to [Schultz et al - 2006]. The following content written is a rephrasing in our own words of the various information we found within this document.

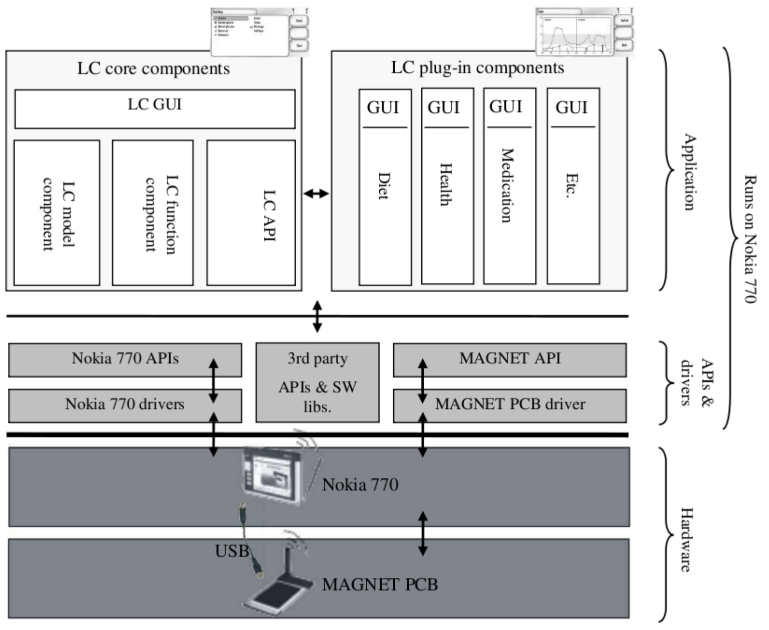
The Magnet.Care framework aims to define a set of scenarios based on the PN solution proposed by the Magnet Beyond project. More precisely, Magnet.Care addresses the overall issue of "Home management of life-style related diseases". The aim of this project is to facilitate the users' management of their disease where and when it is needed. More generally, it provides to users tools allowing them to manage in an healthy way their everyday life. For instance, it provides a tool to create shopping lists... As Magnet.Care rely on the Magnet solution, it will be anywhere the user is located at. This pilot has three main functions:

- To monitor the behavior of the user,
- To compile and present an overview of the monitored parameters to the user,
- To form a basis for the decision making.

2.1.6.1 Magnet.Care: Overall architecture

Relying on the Magnet architecture, Magnet.Care is able to include a large variety of networking devices located in several places. The higher layer of the architecture is the Magnet.Care service which can be viewed as a *Lifestyle Companion*. Below this layer are both the Magnet APIs and drivers needed to enable the communication with the device hardware components.

Figure 2.4 Overall MAGNET.Care Lifestyle Companion design.



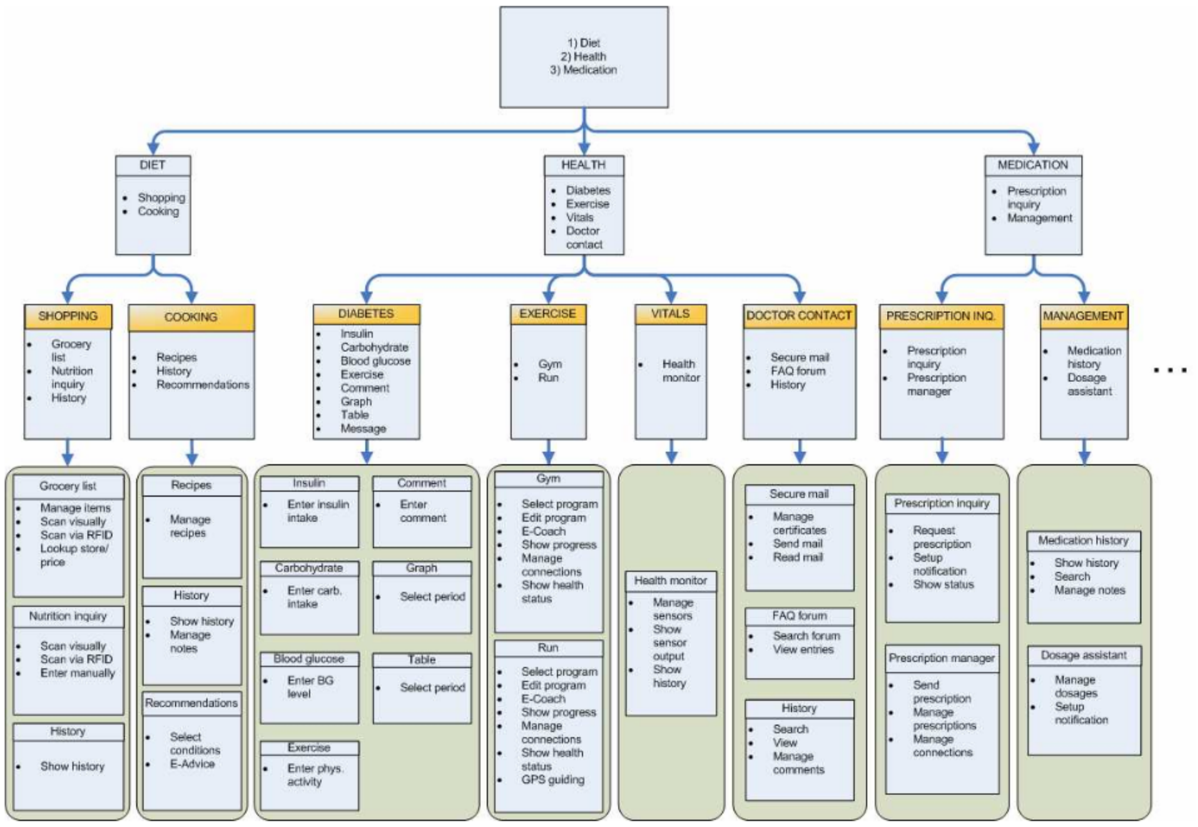
[Schultz et al - 2006] (p16) - This figure shows the high level layout of the MAGNET.Care application. The core Life Companion (LC) can be extended with a number of plug-in modules. The service is built upon the MAGNET API and other device dependent APIs.

2.1.6.2 Magnet.Care: Objectives

The goal aimed by Magnet.Care is to allow the end-user to manage his life-style by providing him a core tool relying on the Magnet framework, which can be extended by adding plugins. Each of these plugins targets a special scenario, and provides function allowing him to fit his requirements.

For instance, the *Cooking* module is aimed to assist the user during cooking time. This module is divided into several functionalities. One of them is the *Recommendations* one. This one is aimed to help the user to select a recipe for cooking. The recommendation given by the system will be based on the user profile (e.g. he needs to eat low-fat meals, . . .).

Figure 2.5 MAGNET.Care Lifestyle Companion framework.



[Schultz et al - 2006] (p18) - This figure shows the framework made for the Magnet.Care pilot service.

2.2 Multi-touch technology and tabletop display

This section refers mainly to the Multi-touch article [3] in [Wikipedia]. The following content written is a rephrasing in our own words of the various information we found within this document.

Multi-touch technology is used to allow users to interact with a machine using natural hand gestures and is at least 25 years old [4] . Tabletop displays usually take advantage of such a technology in order to allow user to interact with documents or applications in a friendly and intuitive way. Usually, these kinds of display are used for collaborative work. For instance, the use of such a tabletop display during meetings allows every user to access the information, and interact with it. But such a setup is not limited to the document manipulation; it can also be used in a lot of new sectors relying on the intuitiveness of the system.

Through this section, we are going to present two multi-touch tabletop products, a commercial and professional one called Surface, and another one on which we are planning to build our project. This

[3] http://en.wikipedia.org/wiki/Multitouch

[4] Beginning in 1982, the University of Toronto and Bell Labs were pioneers in this field of research.

second multi-touch device has been made in the Aalborg university, in a master thesis project called demeTouch.

2.2.1 Multi-touch screens

This section refers mainly to the Multi-touch article [5] in [Wikipedia]. It also refers to [MERL - 2008] to [Fleury - 2007], and to [Larsen - 2005]. The following content written is a rephrasing in our own words of the various information we found within this document.

Multi-touch is a UI [6] technique allowing users to interact without conventional input devices such as a mouse and a keyboard. It usually consists of a touch device such as a screen, a wall or even a tabletop as well as a software stack. This one, at the opposite of a standard input device, is able to get multiple simultaneous points as input. This technique is also more evolved than the classical touch devices is the sense that *single*-touch devices recognize only one touch point.

Multiple input points can be found by several techniques, including for instance: heat, finger pressure, high capture rate cameras, infrared light, electro-magnetic field, ...

Several years ago, Mitsubishi launched several research projects such as [MERL - 2008] and the Freehand Touch Gestures project [7]. These projects demonstrate the growing interest in collaborative work technique. They were concluded by the creation of the DiamondTouch Table which is a multi-user, debris tolerant, touch and gesture activated display that supports small group collaboration. Thanks to it, users can look at each other eye to eye, and because this table is multi-user, everyone can interact at the same time without any need to take turns.

From now on, the multi-touch technology is expected to rapidly become common in more and more devices. This technique is expected to improve users' experience in several domains such as:

- Enhance dining experience by providing services such as food order, refills, bill payment, or even allowing guest to entertain themselves while dining
- Gaming
- Governmental use for instance for military purposes
- Enhance multimedia experience

2.2.2 Surface™

This section refers mainly to the Microsoft Surface article [8] in [Wikipedia]. The following content written is a rephrasing in our own words of the various information we found within this document.

Microsoft Surface™, is a Multi-touch Microsoft product which is a combination of software and hardware technology. This multi-user system allows to manipulate digital content by the use of natural motions, hand gestures, or physical objects.

Microsoft has built this table around four main ideas:

- **Direct interaction** which refers to the user's ability to interact with an application without the need for a mouse or keyboard, but simply by reaching out and touching the interface of the application.

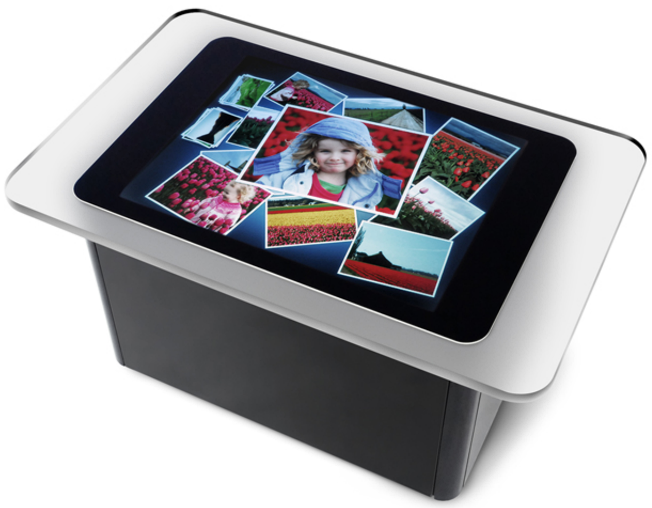
[5] http://en.wikipedia.org/wiki/Multitouch
[6] User Interaction
[7] http://www.merl.com/projects/touchgestures/
[8] http://en.wikipedia.org/wiki/Microsoft_Surface

- **Multi-touch contact** which refers to the ability to have, unlike with a mouse, multiple contact points with an interface.
- **A multi-user experience,** benefiting of the previous idea, several people can interact with an application simultaneously.
- **And finally objects recognition** which refers to the table ability to recognize tagged objects.

To build the designed table, the Microsoft team has used the following technologies to build the table. Surface software technology is mainly based on the Vista operating system. The hardware part is based on a reflective surface placed above a projector and a set of five cameras. These ones are used jointly with a near-infrared, 850-nanometer-wavelength LED light source aimed at the surface. These two components form the computer "vision". When fingertips or objects enter in contact with the table surface, they reflect the infrared light to cameras allowing the table to sense and react to items touching the tabletop. Objects need to be tagged with specially-designed tags if the user wants them to be recognizable by the computer.

This set-up lets the user interact with the machine by either touching or dragging digital objects using their fingers or real-world objects such as paintbrushes. Users can also interact with the machine by placing or moving tagged objects.

Figure 2.6 Surface Computer



The Surface table showing to users a set of photos. These ones are manipulable by users' fingers to easily move, rotate, and scale them. The table is also able to send them over the Internet.

Surface's basic goal is to run applications like photos, music and games in a friendly way for users. Initial customers targeted by Microsoft are businesses such as restaurants, hotels, public entertainment venues and the military for tactical overviews.

2.2.3 DemeTouch Project

This section refers to [Fleury - 2007]. The following content written is a rephrasing in our own words of the various information we found within this document.

The demeTouch project is a 10th semester master thesis project realized by Alexandre Fleury during the 2007 spring semester. One of the project goal was to build a multi-touch tabletop display. This one was built in order to offer the end-user a new approach to listening music. This approach was based on the use of three main senses when using the demeTouch table.

- For the **Sigth**, Alexandre based a part of his project on a continuation of work already done during his studies at the ECE. The work done in this area was based on the Demelo project [9] , which goal

[9] http://projets-etudiants.ece.fr/?/Projets-Pluridisciplinaires-en-Equipe/PPE-2005-2006/DEMELO-Musique-d-ambiance-automatique

was to create a new way to represent music library knowledge gain from the music files themselves, but also by others environmental data such as the listening context. The final idea chosen to describe songs was the color descriptor. This one was based on three different parameters: song genre, dynamic of the track, and acoustic energy. This project was achieved by creating a player based on this color description of songs.

- The sense of **Touch** was used in the demeTouch project when the user wanted to interact with music library on the table surface.

- And last but not least, the **Hearing** was used when the user was listening to a track.

Thus, Alexandre took advantage of the actual table and hardware setup designed by Søren Larsen in [Larsen - 2005] during his master thesis project to build upon this system an application enabling the end-user to navigate through his audio library using fingers in contact with the table surface. This system relied upon four main components:

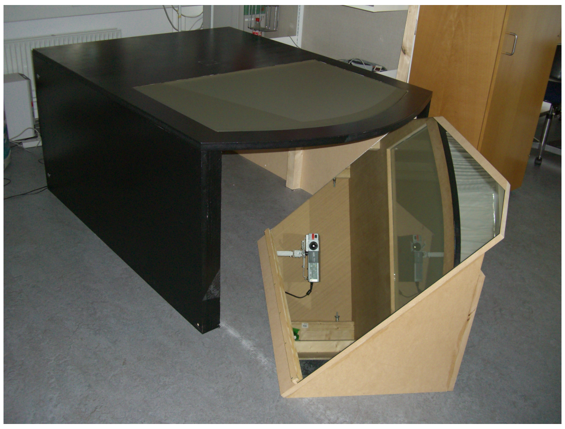
- the tabletop display.

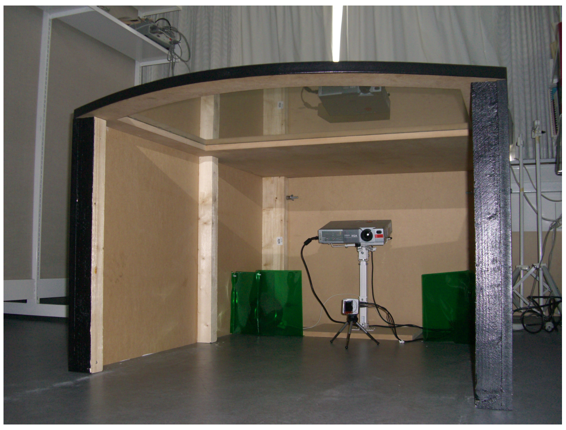
- the video processing engine used to detect fingers on top of the tabletop display.

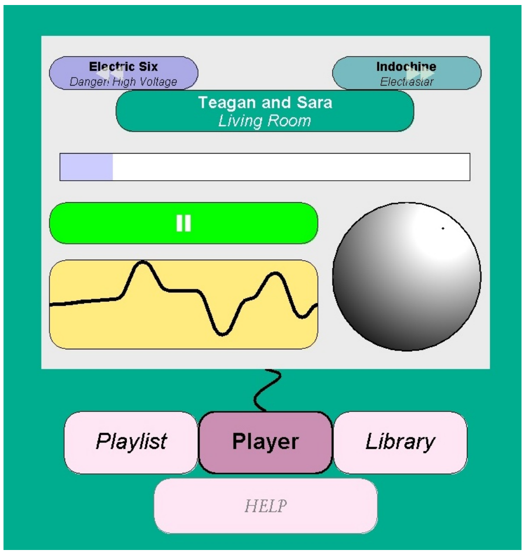
- the colored-music player's processing engine developed by students at the ECE, which extracts music features (genre, energy, dynamics) from audio files and map them to a HSV (hue-saturation-value) color

- And finally the main application which offers an intuitive and pleasant interface, and takes advantage of the previous components.

Figure 2.7 The demeTouch project Table and its interface.


External view of the demeTouch table. The main component we can see on this picture are the mirror and the screen area on the top of the Table.


Internal view of the demeTouch table. The main components we can see from this point of view are the projector, the camera and the two infrared lights.



The demeTouch application running. This is the player screen displaying the different controls available to the user.

The general table design can be found in the Figure 3.6. The basic work done by the image processor is explained in the Appendix A.

2.3 Internet Tablet, smartphones and others communicating devices

Even if we will not use directly any portable devices such as these ones in our project, we are discussing about these kind of devices because they are a big part of the devices targeted by the Magnet project.

Nowadays, more and more devices are able to communicate using their embedded network module. There is also more and more different types of modules depending on the kind of services wanted. For instance, Bluetooth is currently one of the most used protocol to communicate within a short range. This protocol let companies empower their devices by allowing them to use wirelessly other devices providing services. For instance, a Bluetooth hand-free headset for a mobile phone. This kind of protocols (Bluetooth, but also ZigBee, UWB [10] or even Wireless USB) are designed to offer connectivity in small range, usually to create PANs, but this area tends to be reduced to move to BANs [11] . Thus these technologies are mainly used to benefit from a service provided by a really near device. The main devices which currently come with short range module embedded are laptops and mobile phones/smartphones.

On the other hand, more and more devices come also with an embedded WiFi module. This other module lets them benefit from the high speed Internet and all services provided by this communication channel. Up to now, devices which were aimed by this technology were the laptops, but from now on, more and more mobile devices such as smartphones, Internet tablet and also handheld video game console.

Figure 2.8 Nokia N810 Internet tablet

This Nokia N810 is a really small computer. It is capable of doing exactly the same things a computer is capable of: view documents, browse the Internet and use Instant Messengers, view pictures, watch movies and listen to music, ...

2.3.1 iPhone™

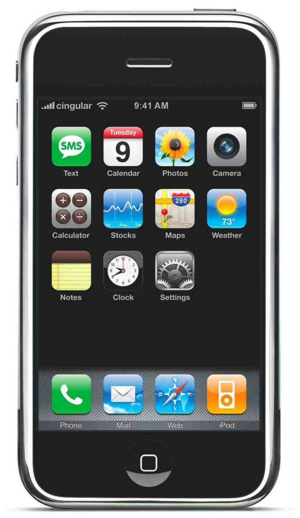
This section refers to the iPhone article [12] in [Wikipedia]. The following content written is a rephrasing in our own words of the various information we found within this document.

Recently, Apple launched a new product named iPhone. The iPhone is more than a simple Internet tablet. This Internet-enabled phone is designed to fit more than simple multimedia requirements. It comes along with a multi-touch screen with virtual keyboard and buttons in addition to including basic functionalities other branded-phones offer. That is to say it includes a camera, a portable music player (iPod) functionality and other Internet related services such as email, web-browsing, and WiFi connectivity.

The "Home" screen shows a list of available applications going from the basic ones: Text (SMS messaging), Calendar, Photos, Camera, YouTube, Stocks, Maps (Google Maps), Weather, Clock, Calculator, Notes, Settings, and iTunes to the four main ones which delineates the iPhone's main purposes: Phone, Mail, Safari, and iPod.

[10] Ultra-wideband
[11] Body Area Network
[12] http://en.wikipedia.org/wiki/Iphone

Figure 2.9 iPhone Functionality



The home menu of the iPhone, showing the main functionality offered by the product. As we can see, if offers a mix of functionalities offered by both a smartphone and an Internet tablet, but with an HCI/GUI designed in a new way, taking advantage of the multi-touch screen.

As all the other Apple branded products, the iPhone HCI is designed in a really user-friendly way. It is able to take advantage of the multi-touch screen to allow users to use their device in a more intuitive manner. For instance, the multi-touch screen allows them to scroll through the library albums just by swiping a finger across the screen. They also can use two finger at the same time to either stretch or shrink the displayable part of pictures, web pages, or even maps. This zooming/magnifying action can be done by pinching together or spreading apart fingertips on the screen.

2.4 Conclusion

Regarding all these new technologies *slowly* entering the marketplace, it clearly appears that networking products such as the one in touch with the Magnet project (presented in the Section 2.1) which are presented in Section 2.3 need to have a user-friendly managing IHM in order to be fully designed in a user-centric way. That is why we want to create an application allowing end-users to manage their PN in a friendly way, tacking advantage of the demeTouch tabletop display realized used and enhanced by Alexandre Fleury during his master thesis.

3 Analysis

In this chapter, we will discuss the different parts of the project, from the user definition to the essential use-cases. We will also present the main tasks we will face during this one-semester project, and a comparison between our ideal system and the real one.

3.1 User definition

First of all, considering that our project is in closed relation to the Magnet project, the users we are focusing on are the one targeted by the Magnet researchers. This means we are targeting users that are keen on having lots of networking new technology devices, in several places, such as computers, smartphones, PDA, and other Internet tablets, cameras, . . . These users are those who are usually the first ones to try new technology out.

Then we think that this kind of technology will not really be affordable for everyone. That is why we want to also focus on owners of public places such as Internet Café, hostels, . . . who offer services to their clients. For instance, an Internet Café client can use the public table to gain access to his PN and be able to manage it. This kind of table could be also found in other public places such as libraries for instance. The only things which will be necessary in this case would be to use this table as a public node in the Magnet sense. This means this table should be accessible to everyone from every PN devices.

3.2 Personal Network management definition

Before introducing the Scenario, the System states, and the Use-cases, we firstly need to describe what managing a PN is.

Regarding the Magnet project, nodes and devices provide services. Managing a PN means from a user point of view, to create or destroy share links between devices within his PN or PN-F. Regarding a whole PN or PN-F, these services are managed by the SMN. Thanks to this node, devices and nodes can run a R&SD[1] to find services accessible to them. Our table will bring to the end-user a really easy solution to create and destroy such links, by only using drag & drop. The following section will introduce the basic scenario which we will use in the next sections to prepare our use-cases.

3.3 Scenario

This basic scenario is going to direct a user willing to enable access to data stored on a mobile phone (for instance, pictures taken with the embedded camera, songs stored in the memory card, phone numbers of his contacts, . . .), from a computer.

[1] Resource & Service Discovery

We use this scenario as a basis for working on the use-cases in the Section 3.5 . Even if this scenario is really a simple one, as our project aimed to simplify the PN user's experience, we want to make all the actions necessary for achieving a task looking alike this description.

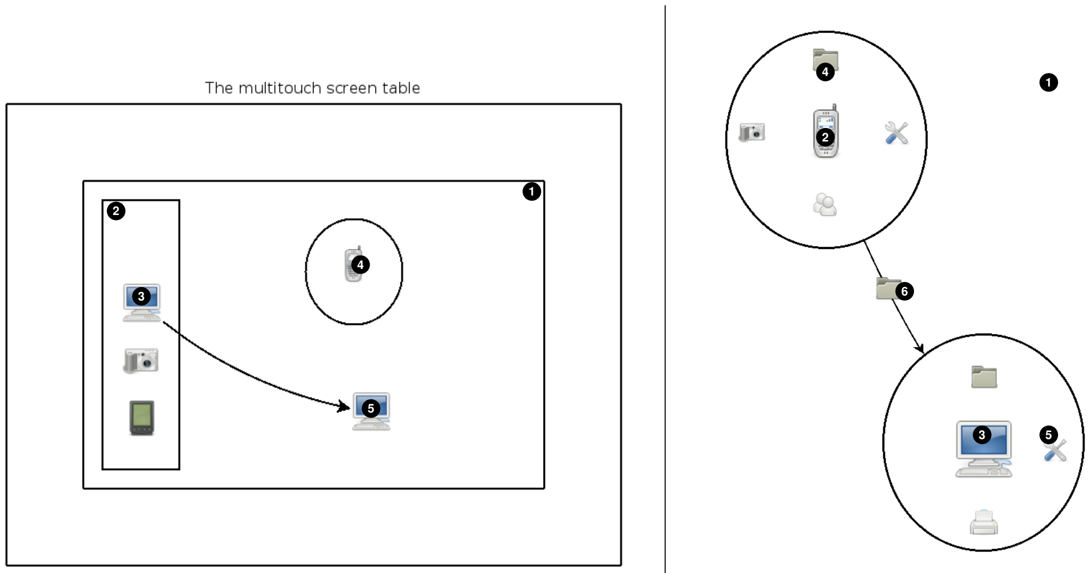
The looking of our application is as we though it at the beginning of the project. The figures shown are thus only the low-fidelity prototype we have realized at that time.

Finally, in this scenario, we are considering either that the user owns the table, and thus the table is already part of his PN, or that he is using an Internet Café public table. In this case, he firstly need to log in and to create a PN Federation. However, in both case, we do consider that the table is already part of a PN or of a PN-F.

3.3.1 The low-fidelity interface

Before the scenario, we need to introduce to the reader our low-fidelity interface. This one is composed of only one "window". This window is divided into two different areas: the *Management Area* and the *List area*. The Figure 3.1 describes all the elements the user can see in our interface.

Figure 3.1 Low-fidelity prototype introduction



❶ The *Management Area* is the main area on the table. Basically, the whole area, except for the *List area*. It is used to display devices selected by the user.

❷ The *List Area* is used to present to the user a list of the devices currently online within the PN, except for the device which are displayed within the *Management Area*.

❸, ❹ Devices are represented by icons in both areas. It is important to notice that the icon is circled within the *Management Area*.

❺ Such an icon which is not circled and which is pointed by an arrow is a dragged icon. It means that the user is currently dragging it from the *List Area* to the *Management Area* to manage its services.

❶ This is a zoom on the *Management Area*.

❷, ❸ Devices are represented by icons centered into a circle.

❹, ❺ Services and resources are represented by icons around the Device one. These ones are movable over other devices to share resources.

❻ A shared resource is represented by an arrow coming from one device (the provider of the service) going to another device (the one which benefit from the share). Moreover, an icon describing the service is shown over the arrow.

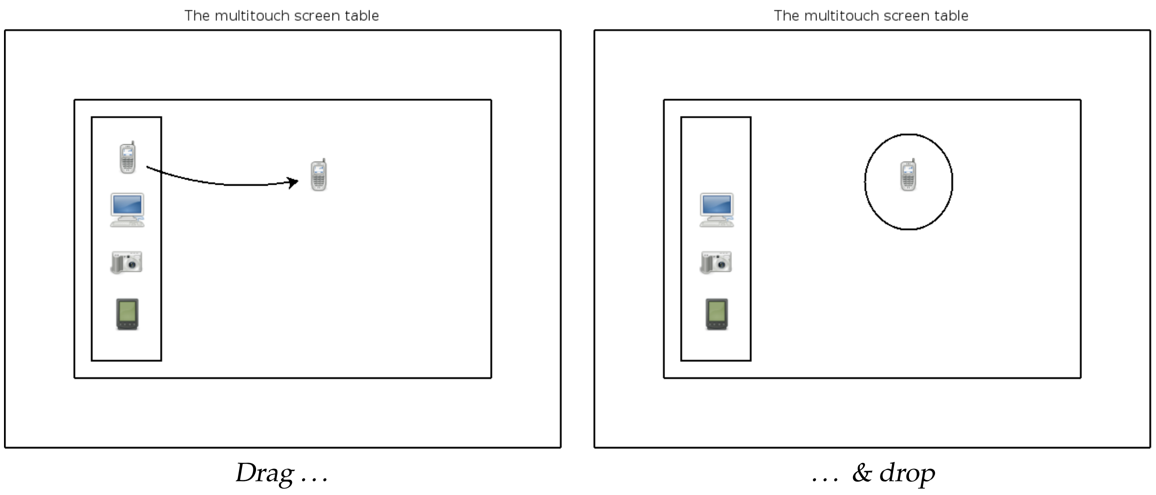
In this interface, the box around the *List Area*, and the circle around the selected devices in the *Man-*

agement Area are just used to differentiate these areas from the main *Management Area*. Moreover, even if a selected device disappear from the *List Area*, it is just a way to show to the user that this devices cannot be at the same time in a manageable state and in an idle one. Thus this device is not removed from the PN although it disappears from the *List Area*.

3.3.2 First step

First of all, the user will go to the table. On the display, he finds on the left the list of the online devices which are part of his PN. As he wants to modify the file access management rights of the phone, he firstly selects the mobile, and "drags & drops" it in the middle of the table which is called the *Management Area*.

Figure 3.2 Low-fidelity prototype 1

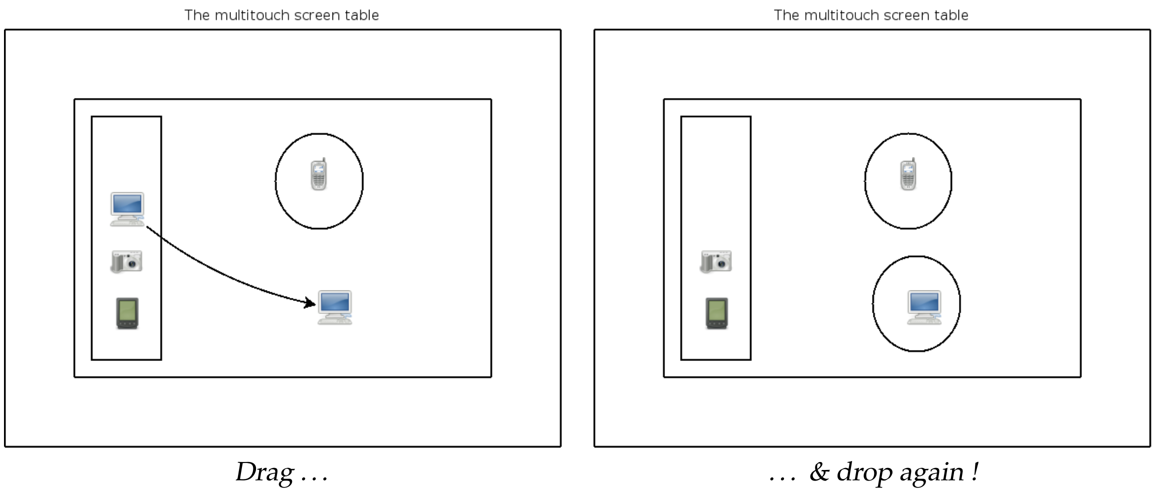


Drag . . . *. . . & drop*

3.3.3 Second step

The device dropped in the *Management Area* is circled. In this circle appear icons which are used to interact with the other peripherals. Currently, the user does not need to use them. He just needs to select the computer from which he wants to access the files.

Figure 3.3 Low-fidelity prototype 2



Drag . . . *. . . & drop again !*

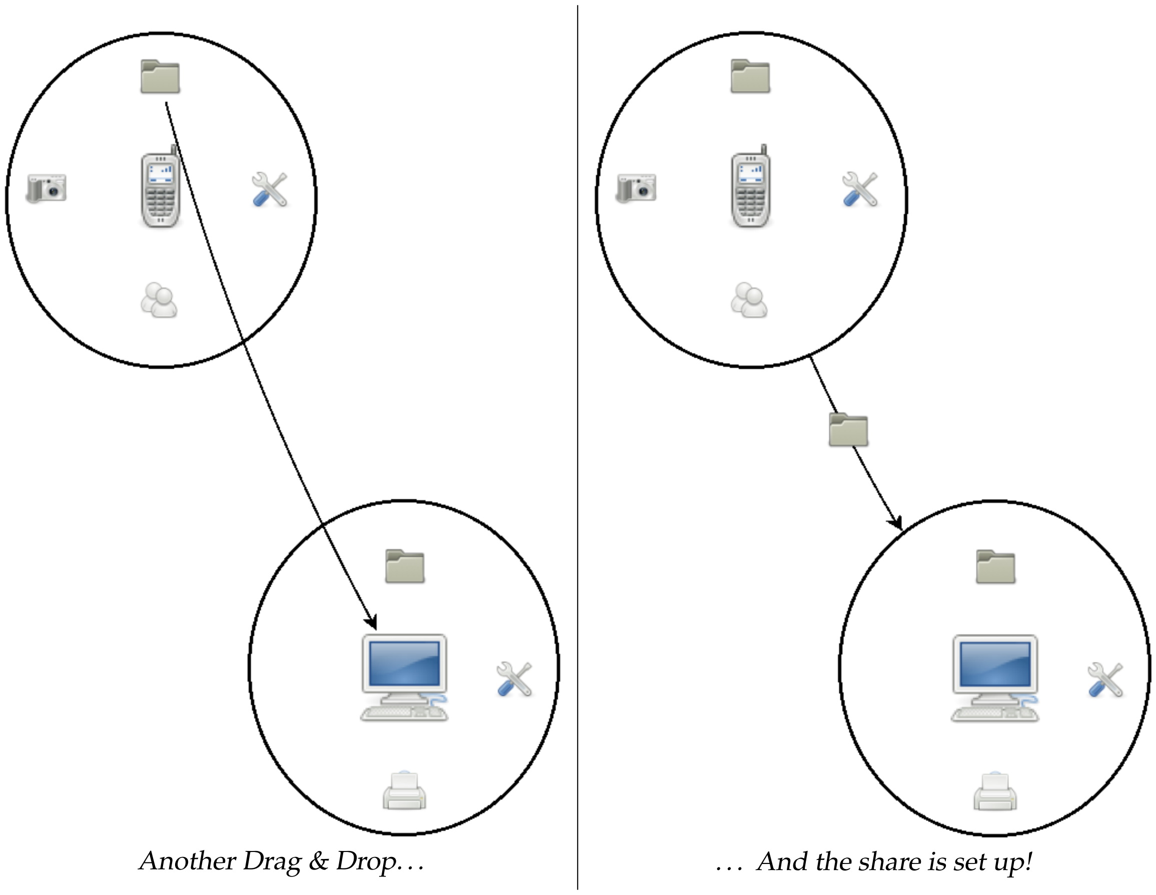
3.3.4 Third step

Now that he has selected the two devices, he is able to manage their relationship. To do this, he just has to *drag & drop* icons from one device to another.

Currently, he wants to share data or information stored on the mobile with the computer. As we just described it, to do this, he just has to drag and drop the folder icon to the computer. This share is shown by an arrow coming from the phone, and going to the computer. The folder icon is shown to inform the user that this link exist.

Finally, this share does not physically move data from one device to another. It only modify the access rights of these data.

Figure 3.4 Low-fidelity prototype 3



Another Drag & Drop... | *... And the share is set up!*

NOTE

There are lot of icons into the two devices area. Each of them have two comportments:

- **Click:** to access small utility softwares or management ones, like sending an email, printing a picture or setting up a device in the PN.
- **Drag & Drop:** to create links between devices.

3.3.5 Last step

Finally, the user is able to bring back devices to the *List Area* in the left part of the table to remove them from the *Management Area*. He only will have to Drag & Drop the device icon into the List area.

3.4 System states

From the last part describing the user actions scenario, we are able to find out several states through which our system will go.

3.4.1 Idle state of the table

When the table is not in use, it basically displays a small device representation list in order to present to the user all his devices or the public ones. To this list might be added some short descriptions of the peripherals in order to make the end-user life easier, but this description will not be always displayed. For instance, it can be a tool-tip which only appears when you stay more than few seconds on an icon without moving.

This list must not be spread on all the display. It should be organized in categories. The user might want to be able to select from the options to order the list either by peripheral type, name, by PN/PNF membership, . . .

3.4.2 The user selects a device

When the user selects a device from the list using his finger, this device should move to the center of the table. This device should appear circled. This circle could be filled with icons describing things which can be managed using the table. For instance, a small folder could appear. This icons means that this peripheral can be used as a storage device. Another icon example is a small screw driver crossing a wrench which means that this device can be configured remotely from this table.

An important thing is that the user is able to select several devices from the list. Using this function should let the user manage interactions between several devices. For instance, if he selects two mobile phones from the list, the standard icons should appear, but moreover, he should be able to move them. Moving the address book icon from one mobile phone to the other allow the user to share the address book of the first phone with the second one.

3.4.3 The user uses the icons to. . .

We think that the icons must be meaningful. That is to say, that they do not need to add some text to describe them. At least, we can imagine having some tool-tips to describe icons and to add them a more precise description; but this text/tool-tip should not be always displayed. It will only appears after spending few seconds above an icon without moving.

Managing contacts, either phone contacts or mail ones should be as easy as touching the Contact icons. This one should be designed to be understood/recognized by every users, coming from every IM/email/. . . client.

Figure 3.5 An example icon well known by every IM user.

The MSN "buddy" icon is often used to refer to contacts in IM client such as Amsn, Meeboo, Pidgin, . . . Thus it could be used to show the address book service in our UI.

Icons should be either clickable or movable.

- Moving an icon from one device to another have a consistent signification. This move means that the user wants to share something between two PN devices. This shared resource can be either data such as (music, pdf, word, . . .) files, an address book, . . . or a service provided by the peripheral (such as a printing capability for a printer, or taking a picture for a webcam enabled device).

Moving an icon will not copy or move any service. For instance, if the user moves the folder icon of a first device above another device, this action will never move or copy any files from the first device to the second one. This comportment is linked to the Magnet concept of granting access to data in the PN and not moving these data.

- Clicking on an icon should also have a coherent signification. It should let the user access an application. This kind of application must be quite simple and provided by the service provider and not by the table by itself.

  For instance, clicking on a printer icon should open a small file manager letting the user select a file from his PN that he wants to print. Clicking on a mail should let him write an email in a small window.

  All these actions should not let the user leave the Table Manager application for another one. If the user clicks on the mail icon, Outlook must not be launched. The user must not have to leave our application for any purpose. Thus our application should handle the display of small interfaces and should also handle sending back commands to the service provider. This way, a device should provide a service: for instance a mail service; plus a description of a remote interface displayable by the table: for instance in XML, and finally, the service provider should handle to get back commands from the displayed interface on the table. This kind of architecture can be compared to the web architecture, in which a server provide a (x)html description of an application, and a browser or any web client can send back to the server commands to execute.

WARNING

Of course, even if such behavior can really simplify user life, it is far from extending the managing role of the table, thus we are not planning to work on such a functionality which should be treated as a whole master thesis project.

3.4.4 The user has finished to use one device

When the user finishes to use a device, he should be able to "remove" it from the main part of the table by a simple "drag and drop" to the device list area. If a device is not used during some time, it should also go back to list area automatically to maximize the selected device area.

3.5 Use-cases and Conceptual Model

3.5.1 Use-cases

In this section, we present the basic use-cases we are targeting. These use-cases try to fit the requirements already described in both Section 3.3 - Scenario and Section 3.4 - System states .

This section refers to Table 3.1, Table 3.2, Table 3.3, Table 3.4 and Table 3.5.

These use-cases come after the realization of the low-fidelity prototype. Thus they globally respect it, but some new features are added. This new features' goal is to make user's life easier by simplifying the UI. For instance, we introduce new icons (eject button for instance) in the service area circled around the device.

3.5.2 Conceptual model

As these use-cases showed it, this project main goal is to simplify user life by letting him manage his PN in a really simple way. That is to say that the end-user will not need to do a lot of things to do a simple task as share a service offered by a device with another device. As the use-cases showed it, the user will need to do only tree actions to make any share-link between devices.

To achieve this goal, our system will need to be build respecting some constraints such as:

Table 3.1 Use Case 0: Logging in a PN

Use Case 0	**Logging in a PN.**
Actor	• PN Manager Table user
Stakeholders	• PN Manager Table • PN infrastructure (PN Agent, GW, . . .)
Scope	• The PN Agent
Preconditions	The user is either: • able to proceed to the insertion of the Table in the PN threw the SMS step currently required by Magnet. • able to directly logging in his PN with his PN number (which is currently the phone number he used to proceed to the PN creation) and his password.
Description: 1. The virtual keyboard appears on the tabletop display. 2. The user "touches" with his finger keys to input his login and validate by touching the "enter" virtual key. 3. The user "touches" with his finger keys to input his password and validate by touching the "enter" virtual key.	
User's criteria of success	• The user get access to the main window of our program; that is to says to the window where are the Management and the List area. • Devices and nodes belonging to his PN or PN-F appear within the List area. This list only shows currently online devices.
Extensions	All the other use cases extend this one and the Use Case 1 in the sense that these ones needs to be done before the other ones. This means that the actors, stakeholders, scope and preconditions are slightly the identical for the other ones.

Table 3.2 Use Case 1: Prepare a device to be manageable

Use Case 1	**Prepare a device to be manageable.**
Actor	• PN Manager Table user
Stakeholders	• PN Manager Table • PN infrastructure (PN Agent, GW, . . .) • PN devices and nodes
Scope	• The whole PN of the user • Public Resources available
Preconditions	• The user has already added the table to his PN. • The devices he wants to manage has to be online within his PN.
Description: 1. The user "touches" with his finger a device in the device list area. 2. He drags it in the management area. 3. And finally, he drops it here.	
User's criteria of success	• The device appears within the management area, and fade out in the list area. • Services belonging to this device appear within a colored circle. This color define the device. One color is assigned to one device, and vice versa.
Extensions	All the other use cases extend the Use Case 0 and this one in the sense that these ones needs to be done before the other ones. This means that the actors, stakeholders, scope and preconditions are slightly the identical for the other ones.

Table 3.3 Use Case 2: Share a service/resource between devices

Use Case 2	**Share a service/resource between devices**
Actor	• PN Manager Table user
Stakeholders	• PN Manager Table • PN infrastructure (PN Agent, GW, . . .) • PN devices and nodes
Scope	• The whole PN of the user • Public Resources available
Preconditions	• The user has already added the table to his PN. • The devices he wants to manage has to be online within his PN. • The user has already selected devices as described in the Use Case 1 .
Description: 1. The user “touches” with his finger a device’s service/resource in the management area, 2. He drags it over another device service list in the management area, 3. And finally, he drops it here.	
Postconditions	• The Table has to send the right commands to nodes, devices and SMN; both within the PN and outside of it.
User’s criteria of success	• The service shared disappears from the colored circle of the device providing the service. It is replaced by an "eject" icon. • The shared service appears in a circle which is a blend of the colors of all the devices which are sharing the service.

Table 3.4 Use Case 3: Unshare a service/resource

Use Case 3	**Unshare a service/resource**
Actor	• PN Manager Table user
Stakeholders	• PN Manager Table • PN infrastructure (PN Agent, GW, . . .) • PN devices and nodes
Scope	• The whole PN of the user • Public Resources available
Preconditions	• The user has already added the table to his PN. • The devices he wants to manage has to be online within his PN. • The user has already selected devices as described in the Use Case 1 , and shared services as described in the Use Case 2 .
Description: 1. The user "touches" with his finger a shared device's service/resource in the management area. 2. He drags it over the "eject" button of one the device which does not need anymore the shared service. 3. And finally, he drops it here.	
Postconditions	• The Table has to send the right commands to nodes, devices and SMN; both within the PN and outside of it.
User's criteria of success	• The service shared disappears from the multi-colored circle of the device who was sharing this service. It is replaced by an "eject" icon. • The "eject" icon in the circle of the device which was not providing the service disappears. The ex-shared service appears back in the circle of the device owning it.

Table 3.5 Use Case 4: End the management of a particular device

Use Case 4	**End the management of a particular device**
Actor	• PN Manager Table user
Stakeholders	• PN Manager Table • PN infrastructure (PN Agent, GW, . . .) • PN devices and nodes
Scope	• The whole PN of the user • Public Resources available
Preconditions	• The user has already added the table to his PN. • The devices he wants to manage has to be online within his PN. • The user has already selected devices as described in the Use Case 1 .
Description 1. The user “touches” with his finger a device in the management area. 2. He drags it drags it back in the device list area. 3. And finally, he drops it.	
User’s criteria of success	• The device fades out and disappears from the management area, and fades in in the list area.

- **Simple interaction solution:** The user must be provided a simple solution to interact with the system. The multi-touch tabletop display is surely a good solution to provide such a interaction solution.
- **Simple Networking solution:** The system must be able to present to the user a list of his personal devices, including the ones which are not physically present in the same place as the table. The Magnet solution to manage such a network of personal devices can provide such tools.

To help users managing their PN, the software running on the table must provide a simple and intuitive interface. This one must show graphically devices, services and resources, without needing to add pages and pages of icons' helper descriptions. The interface does not need any keyboard or mouse to be used. Potentially, the only piece of interface needing to enter some text will be the welcome screen asking to input the PN login and password, but this two fields can be fielded in using a virtual keyboard.

All the networking operation must be totally transparent for the end-user. That is to says actions are equivalent to commands. Each sharing and unsharing actions must be associated to a Magnet command executed to fulfill the user's goal. However, some of the Magnet commands must be totally transparent for the user. For instance, getting the list of devices within the PN or getting the list of services offered by a device must be completely transparent. Finally, when a user action is associated to a command, the application should show a short description to users in order to tell them if the action was correctly reflected to the Magnet personal network or not.

Our application is intended to the user who does know the goal of the Magnet project, and the concept of Personal Network, but does not know how the Magnet project works. That is to say, the user just want one simple solution to fulfill one simple request. For instance, the user's goal can be " *I want to easily share the printing capability of this computer to be able to print pictures directly from this camera.*". All he will have to do will be to select the two devices from the list area, and to select the printing capability of the computer and to drag and drop it on the camera. This is a simple solution to fulfill quite a complex requirement, and that all he wants.

3.6 Basic requirement

The previous use cases illustrate the basic use of our PN table Manager. They provide useful information regarding the design of the user interface, as well as on the interaction design. From these use-cases and all the general description of the system already done, we can define several sub-systems: the physical setup, including the table and its components allowing to detect finger touching the tabletop display, the Magnet node, and our software component. Even if some of these sub-systems have already been introduced in previous sections, all of them are equal in importance regarding the project's issues.

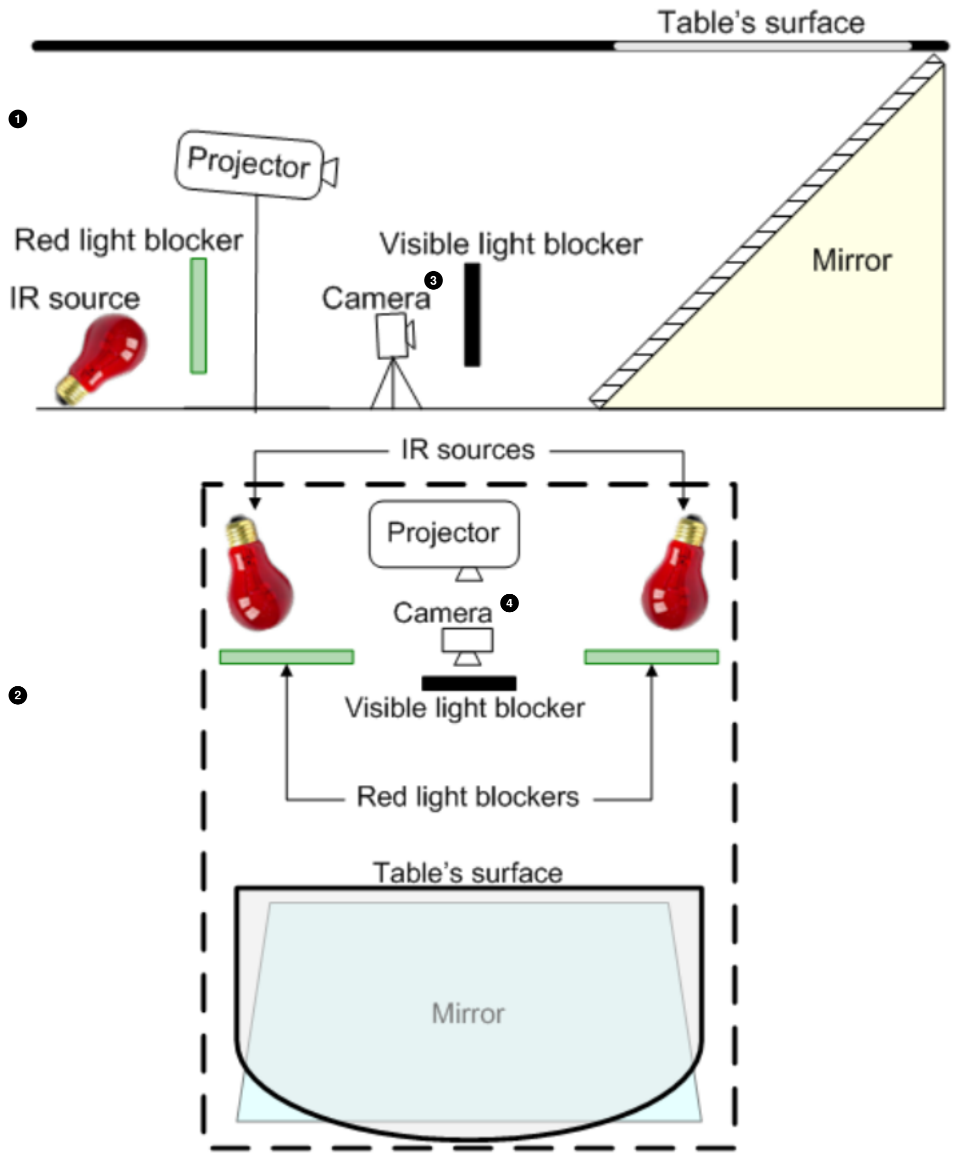
3.6.1 The Table sub-system

This sub-system is probably the most important for the end-user because it is the only one to be "visible". This one has been done by Alexandre Fleury while he was working on his master thesis project. It is composed of two main components: the Table by itself, and the computer running the software stack which is in charge of the video processing engine built to provide a finger detector.

The table as been built to take place in a personal living room, and to be multi-usage. That is why we think this table will perfectly fit the usage requirement we have already describe. On the other hand, the software stack is not as flexible as the physical part. The **fingerDetector** software Alexandre realized for his project is using a java component (JMF [2]) which does not work under a Linux environment. More precisely, the components exists, but it is not able to detect and take advantage of the webcam used to capture images. We are not sure if the problem comes either from a Linux Driver or from the V4L [3] framework, but at the end the result is the same, we cannot use the webcam using a Linux environment. That is to say that we will need to run the finger detector on a dedicated computer running the Microsoft Windows operating system. This computer is backed by a projector and a USB webcam to display capture images of the touchable part of the table. Alexandre has build his finger detector with two main characteristics in mind: speed and precision. These one were needed to ensure both latency-free response from the system, and correct interpretation of the finger position on the screen to avoid erroneous actions of the system.

[2] Java Media Framework
[3] Video For Linux

Figure 3.6 [Fleury - 2007] (p52) Tabletop display Final Setup



❶ Final setup of the tabletop display with infrared system viewed from the side.

❷ Final setup of the tabletop display with infrared system viewed from the above.

❸, ❹ Thanks to a photographic negative placed just in behind the webcam lens, the camera is infrared light sensitive.

3.6.2 The Magnetized component

This component must be run on top of a Linux operating system. This requirement comes from the current implementation of the research project. This implies this sub-system will be running on another computer as the Finger detection component of the table can only be run on top of a Windows box.

The Magnet component will be in charge of all the operation related to personal devices and their associated services. For instance, this module will be in charge of listing online devices in the PN, but also both to search for their (un)shared services and for sending Magnet commands to the SMN in order to apply the new configuration chosen by the end-user.

To conclude, even if this component will not be directly used by the user, this one is really important because the service management relies on it.

3.6.3 Our application software

Our application main goal is to present to the end-user a graphical interface. This one must be designed in a user-centric way. That is to say it must be simple for the end-user to interact with it, thus it must be user-friendly. Moreover, this application should be able to handle several pointers. As there is no easy way to handle multi-touch display both within Java and in standard operating systems such as Linux or Windows, if we want to take advantage of the multi-touch tabletop display made available to us, we will need to handle such table capability within our software stack. As the Table software stack contains a XML-RPC server, this first sub-system will be able to send the finger position to our application using this simple protocol.

3.7 Task analysis and inherited features

In order to design and build such a software fitting with the basic requirements we already discuss in Section 3.6 , we need to build more than a simple application, but a complete system which realizes several tasks. These tasks are the following ones:

- **Join a PN:** This task will allow the Table to access a Personal Network. Thanks to this, our Table will be able to be part of a PN and will be allowed to access the several services offered by the Magnet project. Some of these services are part of our project. For instance, we will need the Service Discovery and Service Management modules.
- **Present a list of PN devices:** Thanks to the Magnet project, and to the membership of our system in a PN, we will be able to access some services allowing us to list of devices belonging to the Personal Network.
- **Service Discovery:** This task will allow us to take advantage of the Magnet project to use several module already developed by researchers. These one will provide us some methods to be able to search for shared services within the PN.
- **Service Management:** This other task will lead us to use several components of the Magnet project allowing us to share and unshare services between personal devices.

  It is really important to understand that our goal is not to move any data (files such as photos, .xls spreadsheet, or even .doc documents), but only to authorize their access from a selected set of PN devices. This set can be either composed of one device, or two, or even more...
- **Designing a multi-touch application:** As there is no simple solution to handle multiple pointers within our application, we will need to handle it by ourselves. Moreover, our application will need to handle the problem coming from the use of a touch screen, such as the lack of precision of the pointers deriving from the geometrical deformation generated by the table setup (the position of the projector, the webcam and the mirror, regarding the display surface.).
- **Enhancing the calibration of the Table:** Currently, the calibration of the Table is hard coded. This solution is not the best one because it is linked to a particular Table physical configuration. Effectively, the distortion and other optical artefacts are related to the position of the camera and the table. We are planning to have a look to the Affine Transform solution to enhance this calibration process. Example of this transformation can be found within the project Wii [4] lead by Johnny Chung Lee, a Ph.D. graduate student from the Carnegie Mellon University. Taking advantage of the Wiimote controller in which is embedded an infrared camera, he is able to track infrared sources such as finger. Thanks to this tracker and the use of the Affine transforms, he built several project such as a a finger tracker, a Multi-point Interactive Whiteboards, or even a Head Tracking system for Desktop VR Displays. The use of the Affine transforms should help us to enhance the calibration process and the overall precision of the Table finger tracker.

Regarding the design of the user interface, we derived from this requirement several features we want to think of in order to make the user life easier. For instance, our application might have :

- **A simple design** to let the user easily learn how to use our software. We are basically thinking to have two main different areas in the application window. There will not be a lot a different windows, because having too may windows might lead to a confusing design.

[4] http://www.cs.cmu.edu/~johnny/projects/wii/

- **No text Menu** will ensure the simplicity of the application by removing from the screen all information which are not really needed.
- **Graphically display information** related to Magnet devices and their associated configuration. This will ensure that our application will be easy to learn, and also that it will be user-friendly. The main lead we are following is the use of a *cluster map*. This kind of map will let us present to a user a node network interlinked by their properties. Thanks to this, a property belonging to only one node will be directly linked to this node. If another one belongs to several nodes, this one will be linked to each node.
- **Handle the touch-screen** function provided by the table to allow the user to interact with our application using only his fingers as pointers.

These features will be discussed and reviewed in the Design part. The main idea is actually to really make the user life easier by thinking the GUI in a user-centered design way. When we worked on defining these features and requirements, we were in the same time working on designing the low-fidelity prototype. This low-fi prototype can be seen in Section 3.3 - Scenario of this chapter. Figure 3.2 , Figure 3.3 and Figure 3.4 show the different screens of the low-fidelity prototype.

3.8 Ideal Vs Real systems

3.8.1 Ideal System . . .

Ideally, our system should be able to take advantage of the Magnet project to easily communicate with devices. As it will be discussed in the Chapter 4 - Design , we will have to use several components such as the Service Discovery and Service Management.

On the other hand, our ideal system would perfectly take advantage of the demeTouch table and its multi-touch capability. That is to say that it should handle the use of several finger at the same time; fingers being either from one or two hands belonging to one or more users.

3.8.2 Real System

However, in the current state of the project, there is no real API. Magnet researches gave us access to the project repository and its wiki. But as there is no real API, we probably will not be able to really use the Service discovery and the service Management components.

The second point is the precision of the finger detector. This one is built using an approximate solution. Thus we will manage to find a solution to enhance this precision. However, the actual finger detector is build upon an image processor which can be quite slow. There is a little bit of latency which should not exist in a marketable product.

Finally, to be usable, the table finger detector need the presence of infrared sources. That is why there are two red light bulbs inside the table. But even with the red light blockers, the sources produces a very intense light which tends to corrupt and alter the finger detector results.

Part II

Choices and Implementation

4 Design

This chapter introduces how the *PN Manager Table* system works: it defines the sub-systems involved and describes the interactions between them. The first section gives the reader an overview of the system's general mechanisms. Then further sections detail each subsystem. The use of the tools defined in the previous chapter is explained, related challenges are exposed and solution are proposed.

The chapter outlines how we thought the system's design before beginning its implementation. This design is the result of the analysis steps done before. The system as it will be described here may not be implemented like this, but this chapter matches one of the stages we have been going through during this semester.

This design is our first vision of the system we want to build. As we discussed earlier, it is composed of several parts, such as the computer dedicated to the table related work, which is the result of the project described in [Fleury - 2007]. Our work is partly running on this system to use the table as a screen. The other subsystem is the Magnetized one which is part of the Magnet PN.

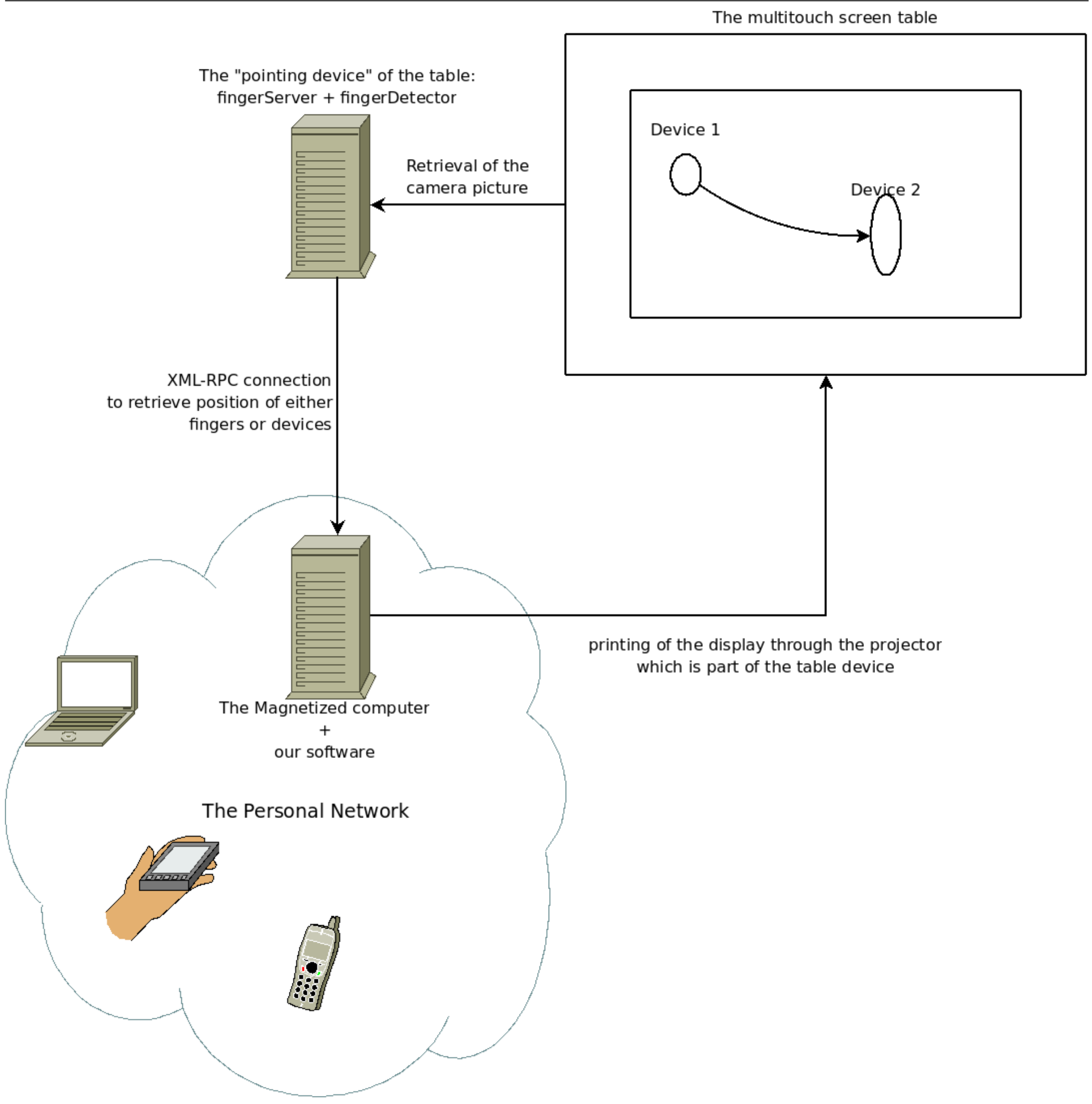
This chapter is divided into four main sections, each of them relating one aspect of the system as we defined it during the analysis part of this project. Firstly, this chapter will introduce the Global architecture section, then it will describe the design of the GUI, the design of the interface between the user and the computer, and finally it will introduce the use we plan to do of several tools such as the XML-RPC communication protocol, and several other tools such as the Affine transform. Finally, it will recap all this design.

4.1 Global architecture / System review

First of all, the table must belong to the personal network in order to interact with it. Basically, the table will be composed of a computer which will be part of one PN, and which might be part of one or more PNF.

This main component will require the use of the table Alexandre Fleury worked on for his master thesis. Currently, this table is a touchscreen. That is to say that the current implementation of the table is made to detect one pointing object on top of its screen. But physically, it is already possible to detect several "fingers". The only thing to modify in order to get a multi-touch screen is the application which will retrieve the data from the table. In fact, we will retrieve from the table data from a XML-RPC connection. Thus, our Magnet software will only have to correctly handle several pointing objects.

Figure 4.1 Basic architecture of system


The multitouch screen table
The "pointing device" of the table:
fingerServer + fingerDetector
Retrieval of the
camera picture
Device 1
Device 2
XML-RPC connection
to retrieve position of either
fingers or devices
printing of the display through the projector
which is part of the table device
The Magnetized computer
+
our software
The Personal Network

Basic architecture of system

Figure 4.1 shows two different main elements: The **fingerServer** and the **Magnet module**. These two modules are shown in two different machines, but ideally they should both run on the same computer. Even though these two modules are different programs, we decided to show them as two computer, in order to properly show the way they will communicate, mainly using XML-RPC.

4.2 Graphical User Interface

This section presents our plan regarding the GUI we are going to design for the end-user. This section will be divided into several subsections. Each ones will introduce a different idea or subject. The first one will introduce the basic composition of our GUI. The second subsection is going to present the Cluster Map representation we have chosen to use to present the PN devices and their services. And finally, the last subsection is going present the main concept the user will be required to understand in order to be able to use our software.

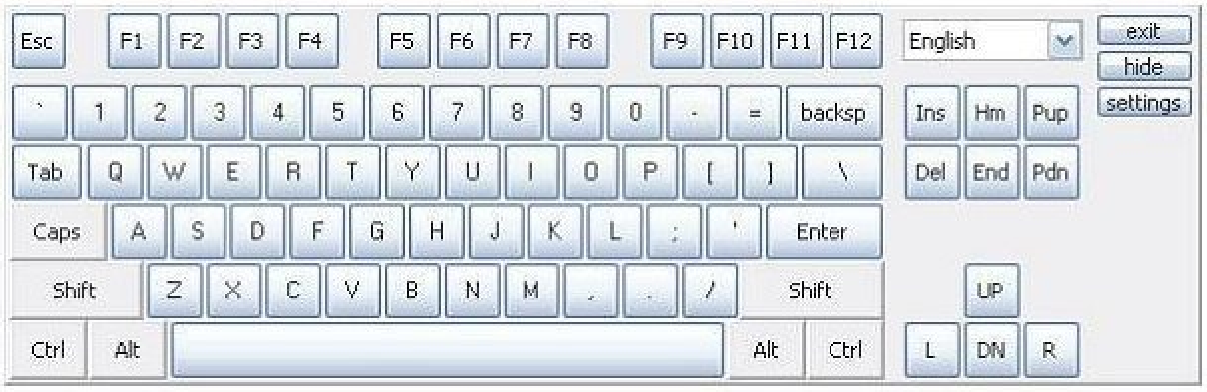
4.2.1 Graphical User Interface composition

In this section, we present to the reader the basic design of our user interface.

4.2.1.1 The Virtual keyboard

The first piece of our software interface the user sees is the virtual keyboard. This one is used by the end-user to enter his PN login and password in order to access to the PN management tool. This interface is basically composed of buttons disposed in the same way as a real keyboard. There are all keys a real keyboard possesses. That is to say, we want to have non-letter keys such as the "Caps Lock" key, the "Enter" key, . . .

Figure 4.2 A virtual keyboard interface example.



Design of a virtual keyboard appearing on a screen.

Of course, such a keyboard is way too complete. We do not need to have special keys such as "Control", "Alt", "Fx" or even the four "arrows". We are planning to only use letter keys, the "Caps lock" to go from lower-case characters to upper-case ones, and finally special keys such as the "Enter" one, the "Backspace" one and finally the "Space" key.

This virtual keyboard will probably appear into the windows asking for the credentials. Thanks to it, the user will be quite at ease when he will be requested to enter his login and password before being able to proceed to the management of PN devices' services.

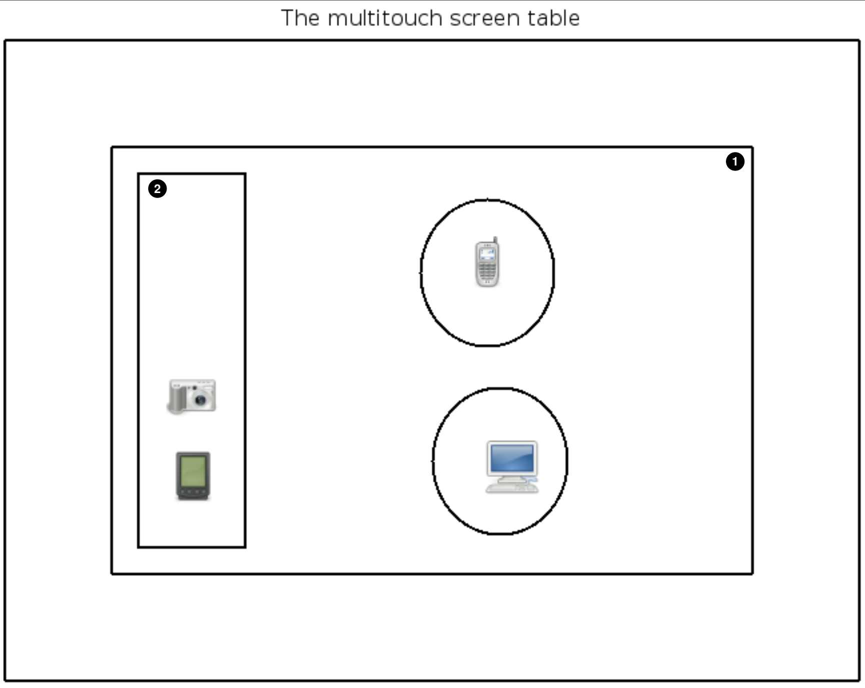
4.2.1.2 The main window

As shown in Section 3.3.1 - The low-fidelity interface , our main window is divided into two parts: the *List area* and the *Management area*. Both of them are used to present to the end-user devices.

- The *List area* presents the devices available in the PN. From this list, the user is able to select some of them in order to manage their services.

- The *Management area* presents to the end-user the devices he selects. Once they are selected into the *List area*, they appear into this second area, where the user sees more details about these devices. For instance, he is able to see both their own services and the ones shared with them.

Figure 4.3 Composition of the User Interface in the Low-Fi prototype.



❶ The *Management Area* is the main area on the table. It is used to display device selected by the user.

❷ The *List Area* is used to present to the user the list of the devices currently online within his PN, except for the devices which are displayed within the *Management Area*.

As already discussed, we are going to remove the box surrounding the *List Area*, because this one can be confusing for the end-user. Moreover, we are also planning to make the selected devices darker than they are when they are not selected, because if they disappear from the first box, the user can think he is deleting them from his PN devices' list.

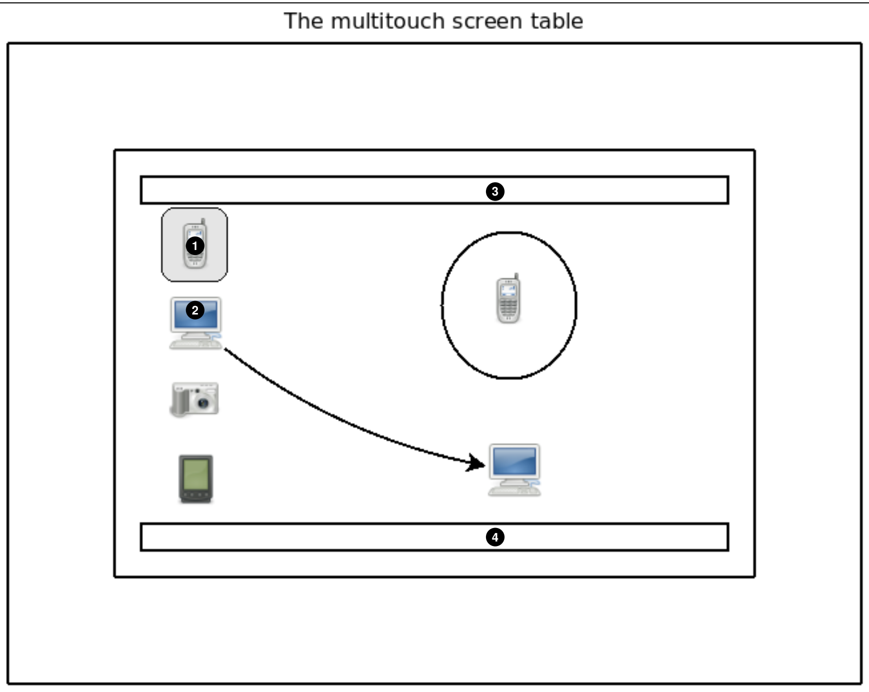
As we think this GUI does not provide enough information to the user, we want to expand this UI by adding two informative panels. The first one will be placed at the top of the screen. This welcome panel will provide information regarding how to use the software for the first time when you never used it. The second panel will be placed at the bottom of the screen. This last panel will be used to provide information to the end-user regarding action he is currently doing. For instance, this panel should be able to tell him:

- From now on, this service is shared with XXXX.

- This service is not shared anymore.

Of course, these messages are only examples. The bottom panel should give information to the user for every actions happening. These messages can be callbacks to inform the user about the outcome of an action, or also tips to tell him why such an action is not doable, and what can he do instead…

Figure 4.4 Design of the Graphical User Interface for the Hi-Fi prototype.



❶, ❷ The *List Area* is no more "limited" (there is no visible border). It contains Devices which are either selected (the grey one) or not (the others).

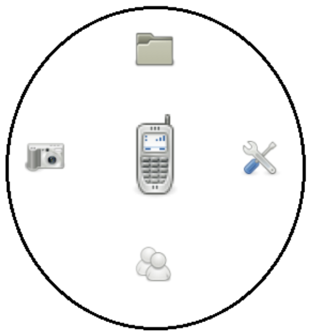
❸, ❹ These two panels are used to present to the user tips, warning or just give him a callback regarding his last action.

4.2.1.3 Devices and Services representation

During the design of the low-fi prototype phase of the project, we were thinking how we would present devices and their services to the user. As shown in Section 3.3.1 - The low-fidelity interface and in the previous section, devices selected will be displayed in the *Management Area*. This is where we need to find a user-friendly way to show graphically to the user links existing between devices and services.

As shown in Figure 4.3 , in Figure 4.4 and in Figure 4.5 , the first thing we thought of when we were designing the low-fidelity prototype was to circle a device, and to show associated services within this circle. We also planned to show shared services using simple arrows. But after having discussed this problem with our supervisors, we realize that that was not a really good idea regarding the user-friendlyness of the interface. The reason what it was a bad idea are multiples. The main one being probably that arrows are not really meaningful, neither beautiful. Having many shared services in the *Management Area* implies having many arrows too; which is really not easily understandable by the end-user. Thus our discussion lead us to find another way to display the services area of a device.

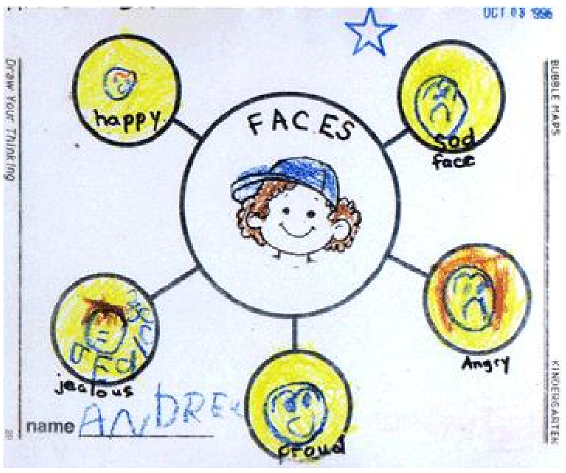
Figure 4.5 A device and its own services



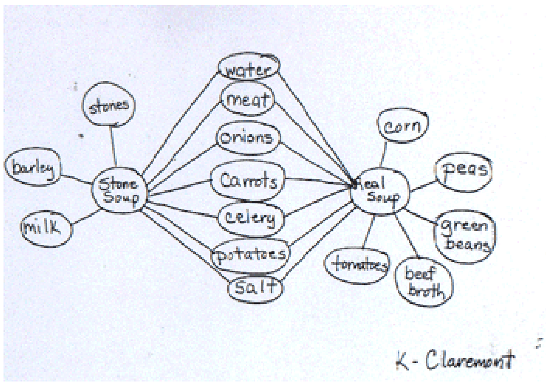
Low-Fidelity prototype design of a device area containing services.

Thus we found other ways to present to the user existing links. For instance, using the (double) bubble map [1] as shown in Figure 4.6 , we could have created one circle per device, and other ones for each services. The idea would have been to link these circles using simple lines or arrows. But it remains one problem with such a map. The more devices and services are available, the more links (and thus arrows) are displayed. Having too many link really complicates the map.

Figure 4.6 Different ways to present to the user devices and their associated services.



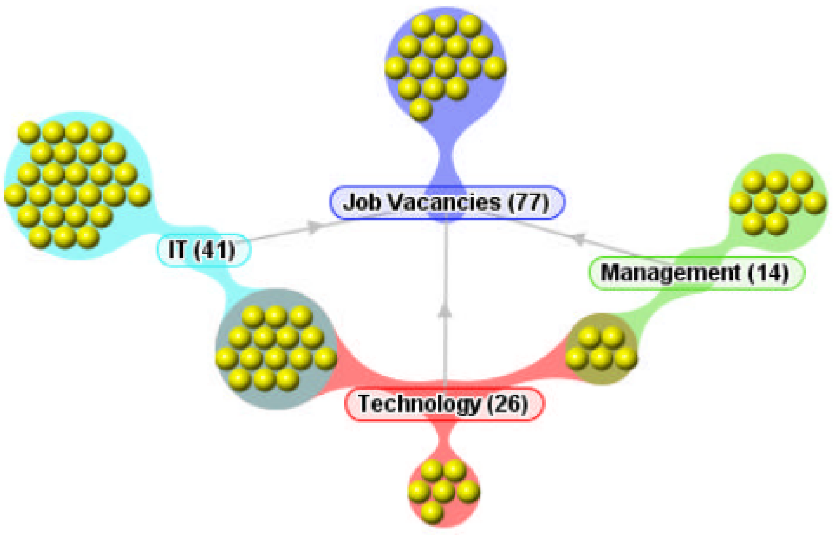
Bubble map example.



Double bubble map example.

Thus we have searched for a new way to present these associations to the user. We found the several ways such as the Euler Diagrams [2] or the Venn [3] ones. We also found in [Spence - 2007] another way to show links between devices based on areas. This method is called *Cluster map*. This is the one we have planned to use.

Figure 4.7 Cluster map example



[Fluit et al - 2005] (p3) - Cluster map example.

[1] http://www.mapthemind.com/thinkingmaps/thinkingmaps.html
[2] http://en.wikipedia.org/wiki/Euler_diagram
[3] http://en.wikipedia.org/wiki/Venn_diagram

4.2.1.3.1 Aduna Cluster map At the opposite of other maps already presented in the last section, Cluster maps are meaningful. Figure 4.7 is an example found in [Fluit et al - 2005] .

> It shows a collection of job offers organized according to a very simple ontology. Each small yellow sphere represents an instance (a job offer). The classes are represented as rounded rectangles, stating their names and cardinalities. Directed edges connect classes and point from specific to generic (e.g. IT is a subclass of Job Vacancies). Balloon–shaped edges connect instances to their most specific class(es). Instances with the same class membership are grouped in clusters. Our example contains six clusters; two of them represent overlaps between classes.
>
> [...]
>
> The added value of our visualization lies in its expressivity. The classes and their relationships (the vocabulary of the domain) are easy to detect. Also, it is immediately apparent which items belong to one or multiple classes, which classes overlap (e.g. Technology and Management) and which do not (IT and Management). The cardinality of classes and clusters is visible: the top class has the most objects, Technology shares more objects with IT than with Management. The subclasses of the root class are incomplete as their union does not cover the superclass: some members of Job Vacancies were not further classified.
>
> — [Fluit et al - 2005]

Aduna Software [4] provide a Java API allowing us to create such maps for visualizing sets of classified objects. Its main purpose is to show if and how these sets overlap, very similar in nature to Venn diagrams and Euler diagrams. Thanks to it, we are planning to create classes (such as "IT", "Technologies" and "Management" in Figure 4.7) for devices. Services will be instance which will belong to one or several classes (when a service is shared between several devices). And finally, thanks the API, instances with the same class membership will appear grouped in clusters.

4.2.2 Drag & Drop

As already discussed in the Section 3.3 - Scenario , the main concept required to be understood by the user to be able to use the PN Table Manager will be the Touch concept. Thank to the **FingerServer** component of Alexandre's demeTouch project our software will be able to handle the tabletop touch screen. That is to say that our program will retrieve finger position thanks to XML-RPC (discussed in Section 4.4). Being aware of the finger position, our application will be able to handle the user request.

The design of this interface allowing the user to interact with the computer will be discussed in Section 4.3.1 .

Thanks to this concept, and as already stated in the use-cases in Section 3.5 the user will be able to manage his PN by only dragging items and dropping them in special area. That is ta say that every items related to the PN will be movable. These items will also be special area where the user will be allowed to drop other items.

This drag and drop concept can be associated with two main actions. The first one is the selection action to allow the user to select one devices in the *List Area* to get more information regarding its services (both own services and shared ones) in the *Management Area*. The second association links the drag and drop to the share/unshare action. These two commands let the user modify the internal association between services and devices within his PN/PN-F.

4.3 Human-Computer Interface

In this section, we are going to discuss the design of the HCI part of the project.

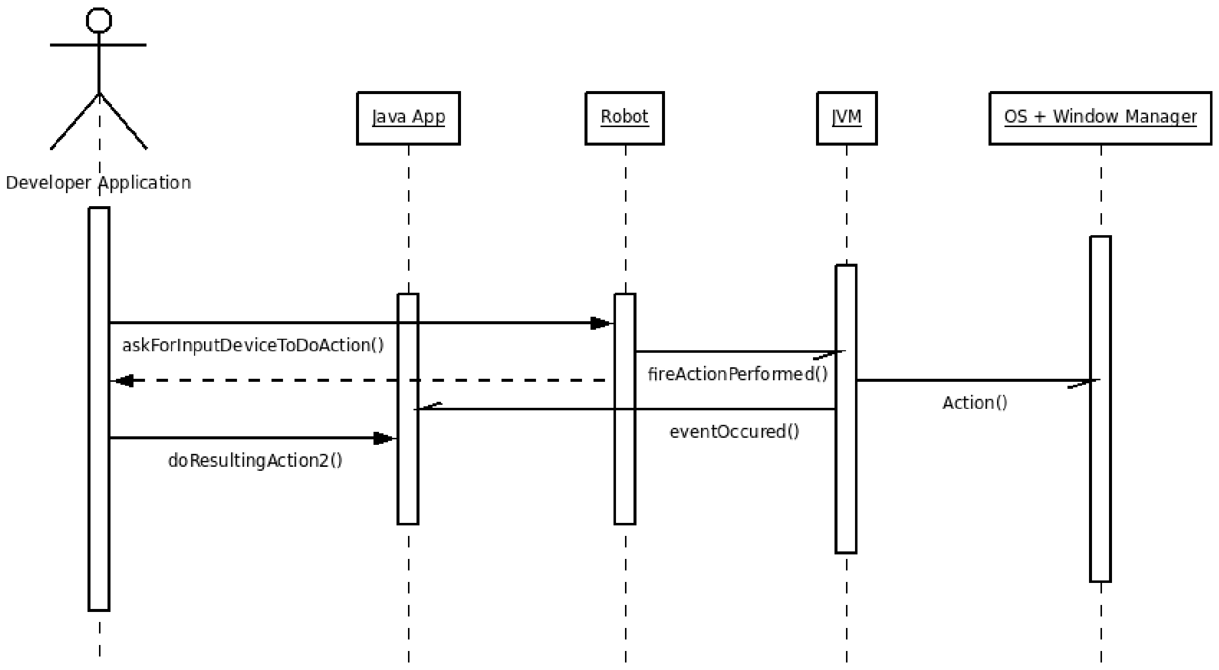
4.3.1 Robot

To allow the user to interact with the PN Table Manager, we are going to take advantage of the **Robot** Java class. This one allow the programmer to simulate the use of standard input devices such as a keyboard and a mouse.

[4] http://www.aduna-software.com/technologies/clustermap

Such a class let the developer handle input devices such as the mouse regarding another kind of input already handled by the developer and received by the software. In our system, this other kind of input is the finger coordinates received from the **FingerServer** module of Alexandre's project implementation.
To easily understand how such a **Robot** works, the reader can refer to Figure 4.8 . This figure presents the relations existing between the alternate input device which is in our system the finger coordinates, and the **Robot**.

Figure 4.8 Sequence diagram: Robot



Sequence diagram describing the functioning of the Robot in our system.

4.3.2 Virtual Keyboard

The virtual keyboard is an important interface between the user and the computer, because the first thing the user must do to be able to manage services-devices association within his PN is to enter his PN login and password. To achieve this required task, we have designed a virtual keyboard interface within Section 4.2.1.1.This one is only a visual interface between the user and the system. Basically it takes advantage of the **Robot** discussed in the last subsection in order to allow the user to "press" keys. This action is done by touching the letter on the tabletop display. It will be observed by the **fingerDetector** and will be sent to the **Robot** via the **fingerServer**.

4.4 XML-RPC

XML-RPC [5] is a remote procedure call protocol which uses XML to encode its calls and HTTP as a transport mechanism. As every other RPC [6] protocols, it is part of the *Application layer* in both the OSI [7] model (layer 7), and in the TCP/IP [8] one (layer 5). We are planning to use it in order to enable all our subsystems (the **fingerServer**, our Application, and the Magnet component) to communicate with each other.
For instance, a program A calling the method *methodB()* registered in the XML-RPC server B will ask B for the (local) execution of the method *methodB()*, which will return over the network the result to A.
The following is an example of how we are planning to use the XML-RPC connection to enable the **Robot-fingerServer** communication.

[5] http://en.wikipedia.org/wiki/XML-RPC
[6] Remote Procedure Call
http://en.wikipedia.org/wiki/Remote_procedure_call
[7] http://en.wikipedia.org/wiki/OSI_model
[8] http://en.wikipedia.org/wiki/TCP/IP_model

Basically, the **fingerDetector** currently has a module called **fingerServer**. This one is XML-RPC based server. That is to says it has registered several methods such as *getFingers()* or *testServer()*. As these methodss are registered, any XML-RPC client can connect to the server using the right IP and the right port number, and call these method. As every method in every program, they will return data as if they were defined and called in one same application.

4.5 Affine Transform

This section is going to introduce the general design of the Affine Transform tool and their use in our application.

As a reminder, Affine transform will be used in the calibration process in order to both enhance and simplify the demeTouch implementation of the calibration process.

In geometry, an affine transformation between two vector spaces consists of a linear transformation followed by a translation: $x \mapsto Ax + b$

Physically, an affine transform is one that preserves:

1. Collinearity between points, i.e., three points which lie on a line continue to be collinear after the transformation
2. Ratios of distances along a line, i.e., for distinct collinear points p1, p2, p3, the ratio | p2 - p1 | / | p3 - p2 | is preserved

[...]

Affine transformation of the plane To visualize the general affine transformation of the Euclidean plane, take labeled parallelograms ABCD and A′B′C′D′. Whatever the choices of points, there is an affine transformation T of the plane taking A to A′, and each vertex similarly. Supposing we exclude the degenerate case where ABCD has zero area, there is a unique such affine transformation T. Drawing out a whole grid of parallelograms based on ABCD, the image T(P) of any point P is determined by noting that T(A) = A′, T applied to the line segment AB is A′B′, T applied to the line segment AC is A′C′, and T respects scalar multiples of vectors based at A. [If A, E, F are collinear then the ratio length(AF)/length(AE) is equal to length(A′F′)/length(A′E′).] Geometrically T transforms the grid based on ABCD to that based in A′B′C′D′.

Affine transformations do not respect lengths or angles; they multiply area by a constant factor: area of A′ B′ C′ D′ / area of ABCD.

— wikipedia.org [9]

Affine Transforms are a mathematical tool to map every point in a vector space into the corresponding one in a second vector space. That is to say it can help us to solve the distortion problem introduced by the table design. Effectively, as the web cam and the projector are not exactly at the same place, a small deformation is introduced into the mapping of the projector's coordinate regarding the webcam ones.

The general design idea is to use the Affine transforms to solve this problem. To achieve this, just after starting the program, the first user (only him) will see appearing a blank screen on which small point will appear. He only will have to touch them. Thanks to this, the computer will know the coordinates of the finger in both the projector plan (the coordinate given to the program to display the square), and in the webcam one (the coordinates retrieved through the XML-RPC connection), and will be able to construct the translation matrix needed to the Affine transform.

4.6 Magnet

The Magnet subsystem will simply be a magnetized computer. Such a computer is able to communicate with all the PN nodes and devices. It will be able to communicate with the SGN [10] which is in charge of managing the services. To achieve this task, the node will need to use the SMP [11] . Basically, our

[9] http://en.wikipedia.org/wiki/Affine_transformation
Other graphical example of affine transform can be found at http://www.glyphic.com/transform/

[10] Service Gateway Node

[11] Service Management Protocol

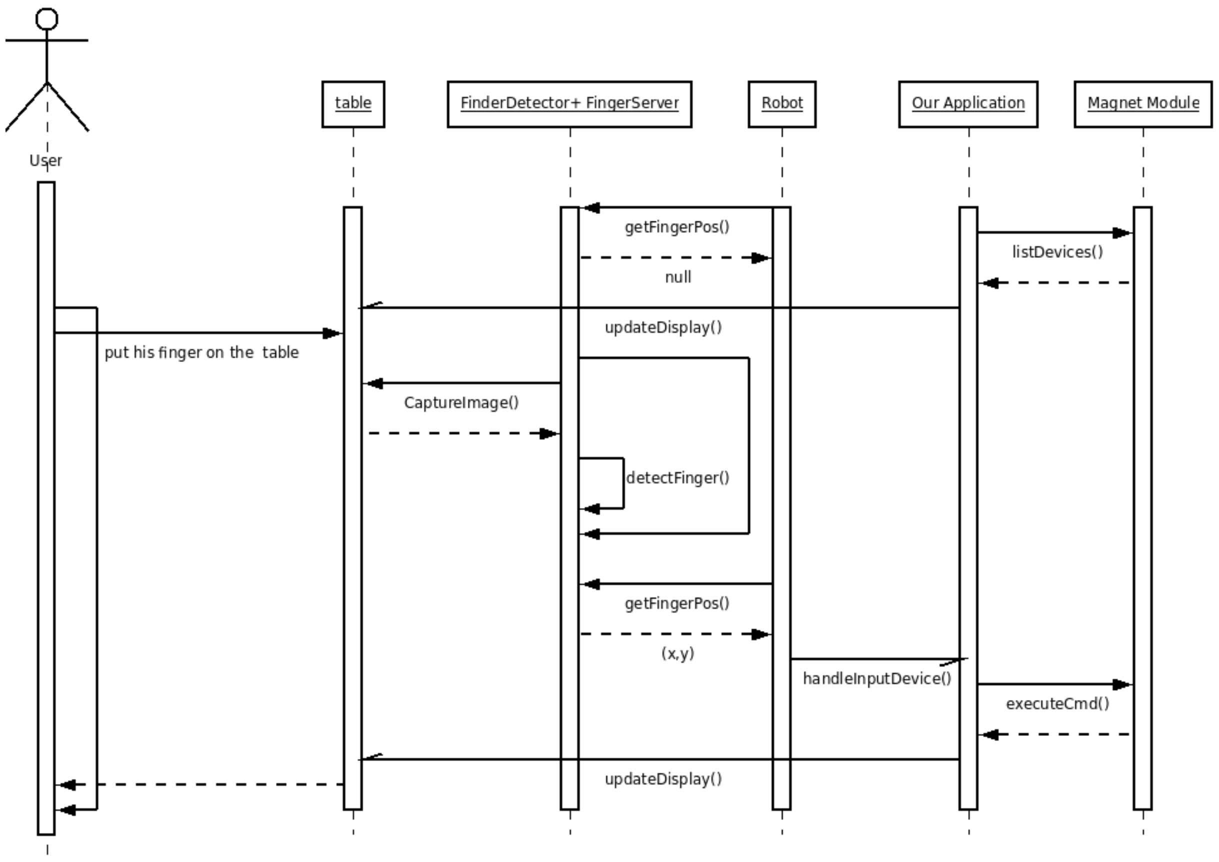
magnetized component should run the UCL [12] plus several other Magnet modules such as the MSMP [13] , and the SCMF [14] which are used to manage services regarding the context. Thanks to these Magnet modules, nodes could communicate securely using their encryption/decryption keys. As a reminder, a Magnet PN ensure the privacy and the confidentiality of the communication within the private network overlaying public infrastructures such as Internet.

Theoretically, Magnet should come with an API allowing developer to perform service discovery and service management. Thanks to this API, we should be able to create a XML-RPC server handling the basic Magnet commands. These methods should therefore be called from any XML-RPC client.

4.7 Summary of the system design

Thanks to all the previous modules already defined, we can build a complete system such as the one shown in Figure 4.9. Such a system reacts to user inputs (finger touching the table) thanks to the **Robot** module which can lead our application to send command to the Magnet component.

Figure 4.9 Sequence diagram: whole system



Sequence diagram describing the functioning of the whole system.

[12] Universal Convergence Layer
[13] Maget Service Management Platform
[14] Secure Context Management Framework

NOTE

Regarding Figure 4.9, the way the **Robot** is working can be a bit strange for the reader. In fact, it is working continuously in a thread as a separate application which is only polling the **fingerServer** for finger coordinates. That why once the **Robot** is working, it began polling the **fingerServer**.

5
Implementation

After presenting the system in Chapter 4, this chapter is intended to give the reader an overview of the changes applied to our original design, as well as to give an overview of the problems we have encountered during the development phase. It describes the main components of our system and presents some technical details concerning their implementations.

Each of the following subsection contains an UML class diagram of the corresponding Java package. These diagrams contains mainly the classes' descriptions. If the readder wants to get more information about these packages or the classes, he can refers to the JavaDoc we have generated from our source code.

5.1 Selection of the implementation language

We chose to develop our application using the Java language because we are used to write programs in this language. That means that we did not loose time learning a whole new development language, as well as new conventions while writing the code.

Moreover, the API's we use for our application were all written in Java. While some of them could easily be found for other development languages, the core of our graphical interface relies on an Java API.

Last but not least, we had access to some classes and methods written in Java for the Magnet project.

5.2 Interaction using XML-RPC and the Robot class

As it has been described in the Design part of the report, we retrieve the coordinates of the user's finger on the table via an XMLRPC connection.

That being said, we still need to use these coordinates. What we wanted to do was to simulate interaction between the user and the application the same way a mouse would interact with it. This means, being able to click, drag and drop elements of the application.

The first part is obviously to retrieve these coordinates. We wrote a class, called Client.java, whose job is solely to do so. By using classes from the *org.apache.xmlrpc* package, we were able to instantiate a XML RPC client object and give it the proper configuration :

Example 5.1 Sending data to the server

```java
XmlRpcClientConfigImpl config = new XmlRpcClientConfigImpl();
try {
    config.setServerURL(new URL("localhost:9966"));
} catch (MalformedURLException e1) {
    e1.printStackTrace();
    System.out.println("error in the server URL...");
}
XmlRpcClient client = new XmlRpcClient();
client.setConfig(config);
```

Since the client should always be running, the *Client.java* class implements the Runnable interface, which means it can be run in a separate thread, hence running in the background of the application. To retrieve the coordinates, we have to send the correct request to the finger recognition server. As a result, the server sends back a string containing the coordinates of the fingers (if there are any). Then we have to parse this string to get the relevant information.

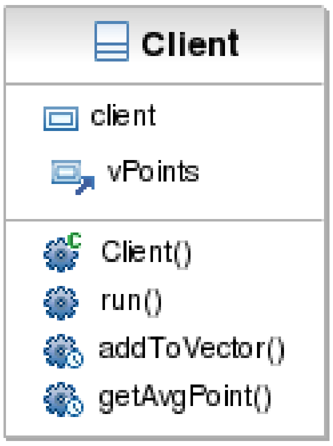
This is only the first step: we still have to use these coordinates to interact with the application. Since our application uses Swing components, the best way to interact with the components is to simulate the use of a basic mouse. That is why we use a Robot object (*java.awt.Robot*):

> This class is used to generate native system input events for the purposes of test automation, self-running demos, and other applications where control of the mouse and keyboard is needed. The primary purpose of Robot is to facilitate automated testing of Java platform implementations.
>
> — Sun Java API [1]

The only problem is, how can we know when the user touches a component in our application, when does the drag starts, and when does the drop occurs?

While trying to find a solution to this problem, we came with the idea of storing not only the last coordinate found in a variable, but actually storing the last 10 coordinates in a vector. If the vector is empty, then the user is not interacting with the table. Once he/she starts touching the table, the vector is filled with the coordinates, in a first-in first-out way. When the user stops, the vector starts filling with empty coordinates. What we decided is, as long as the vector is not empty, a drag action is occurring, and the initial position of the drag action is where the first coordinate was. When the vector is entirely empty, we consider that a drop action occurred at the last coordinate retrieved.

Figure 5.1 The XML-RPC client class-diagram.



The XML-RPC client class diagram. This client works in a thread to continuously poll the fingerServer. Concurrently to this thread, the Robot entity continuously retrieve the finger position to reproduce the user's behavior using the mouse pointer.

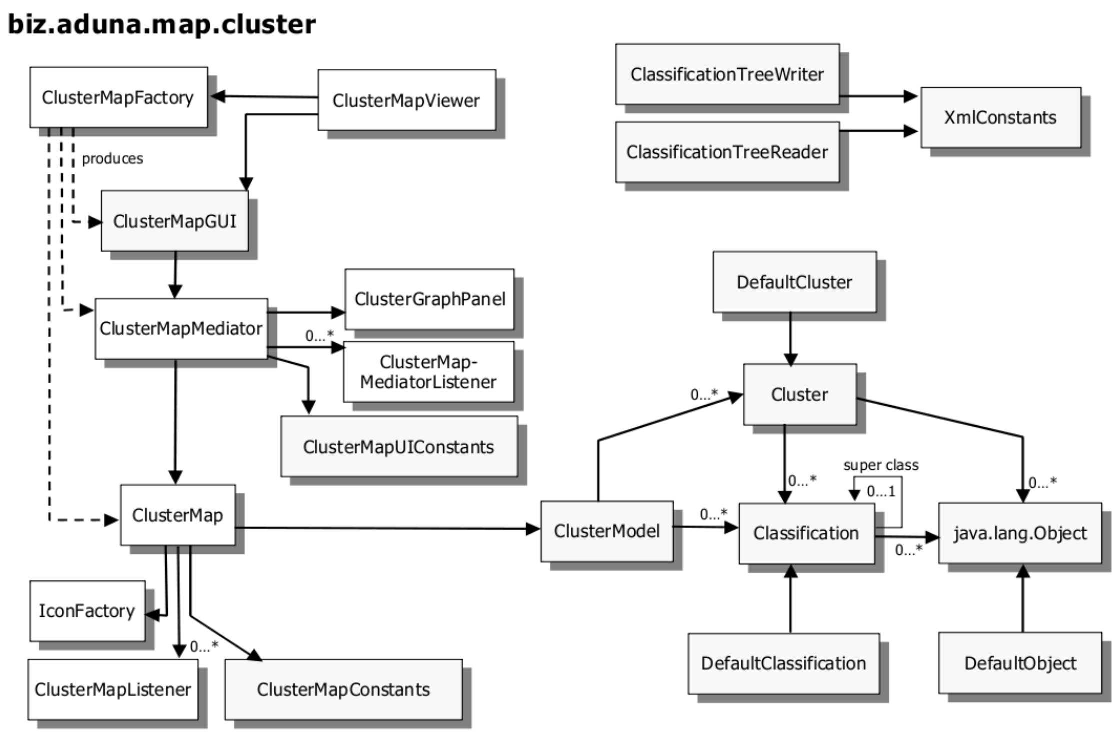
[1] http://java.sun.com/j2se/1.5.0/docs/api/

5.3 Cluster Map and Core Elements

In order for our application to have the closest look to our prototype, we use an API called **clustermap**, which is developed by Aduna [2] under GNU GPL V3.

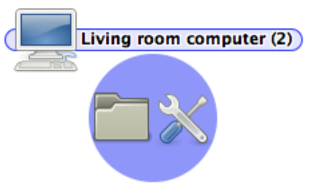
Using this API, we are able to create graphs that represents devices and their respective services. A device is represented by a cluster, and within this cluster the services can be found.

Figure 5.2 Aduna Clustermap API class diagram



Aduna Clustermap API class diagram.

Figure 5.3 A Device representation



Representation of a PN device thanks to the Clustermap API.

In a developer's approach, a device is represented as a classification. That means that services can be *classified* under a device. A graph displays the classifications that are selected.

The cluster map API is as follow:

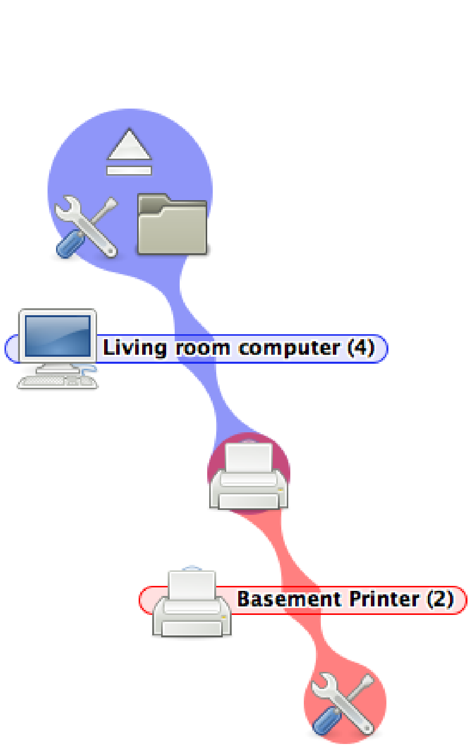
As the diagram in Figure 5.2 shows, the **ClusterMapFactory** object is used to create a **ClusterMap-Mediator**. The latter is a key object, it is the one which creates and handle updates of the a **Cluster-GraphPanel** (i.e. the graph).

We also wrote a class which specifies which icon correspond to which device/service. This class implements the interface **IconFactory**, and the implemented methods allow us to display the correct icon for a given device or service.

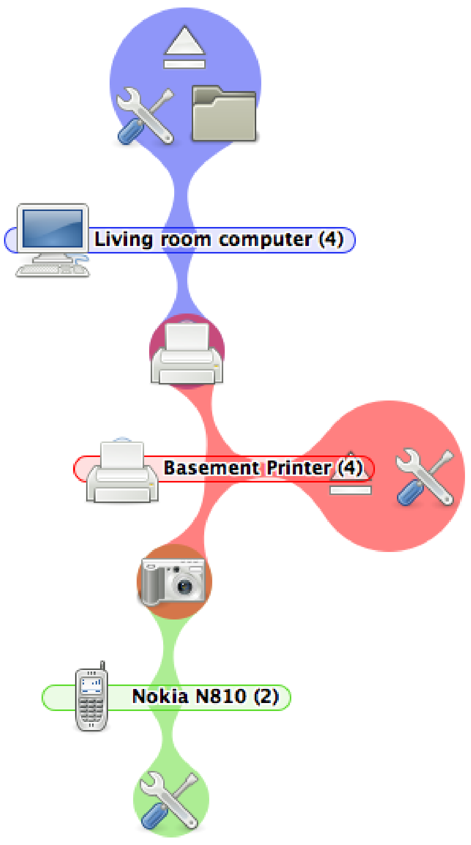
This represents only the graphical representation of devices and services. The graph, by itself, does not handle drag and drop actions. This is why we are using another package, that we chose to call **Ghostpane**.

[2] http://www.aduna.biz

Figure 5.4 Different ways to present to the user devices and their associated services. (Hi-Fi prototype)

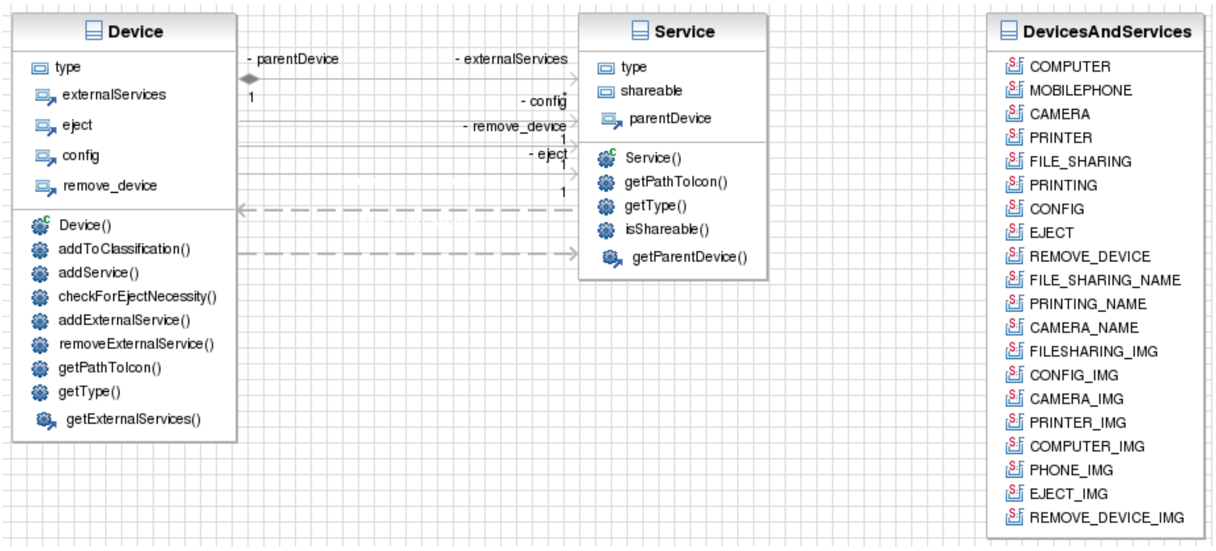


Two devices sharing one service.

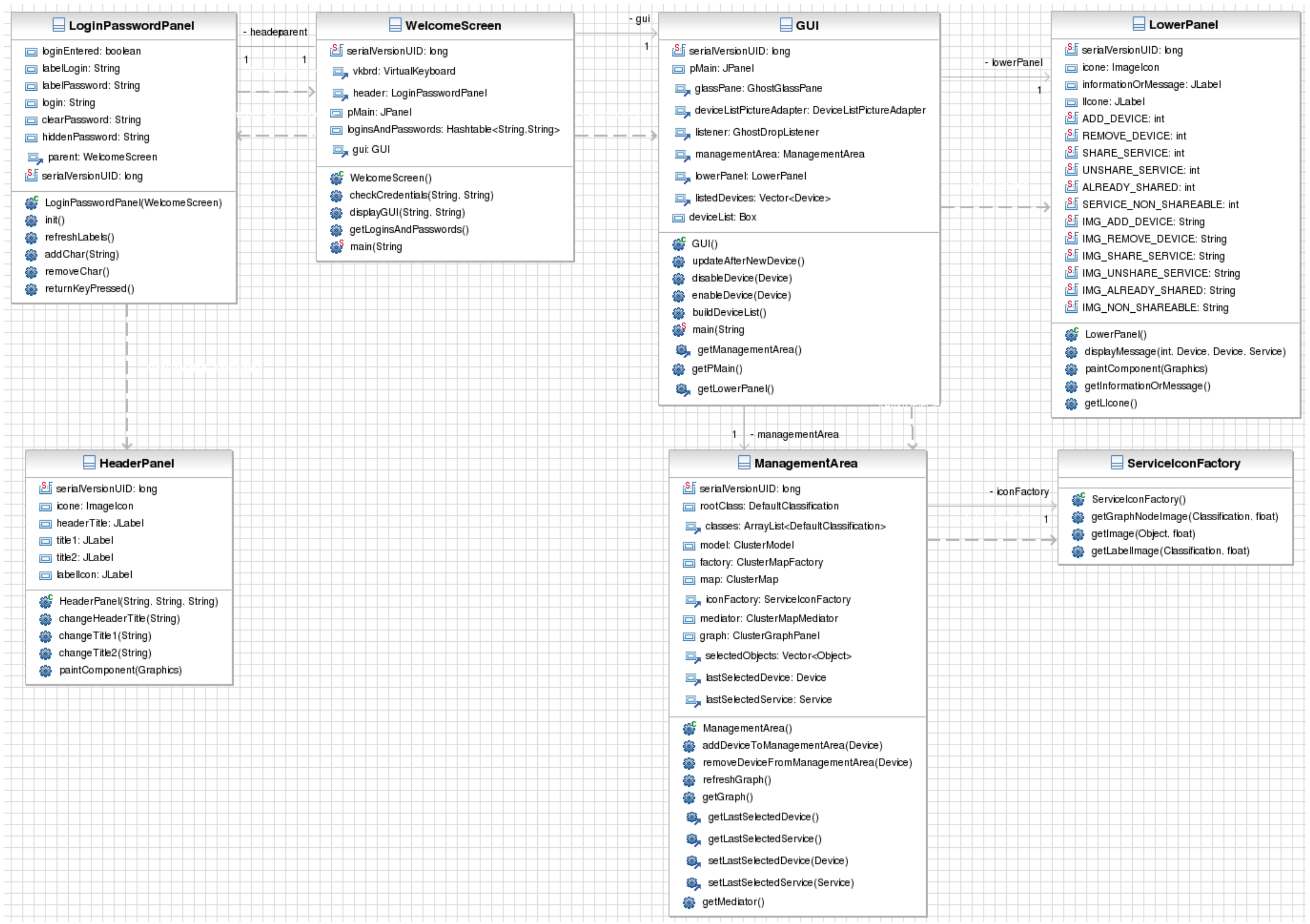


Customs sharing organization.

Figure 5.5 The Core class diagram



Class diagram for the Core package. The DevicesAndServices class is used to store all the definitions for devices and services used in the other classes.



Class diagram for the gui package.

5.4 Ghostpane

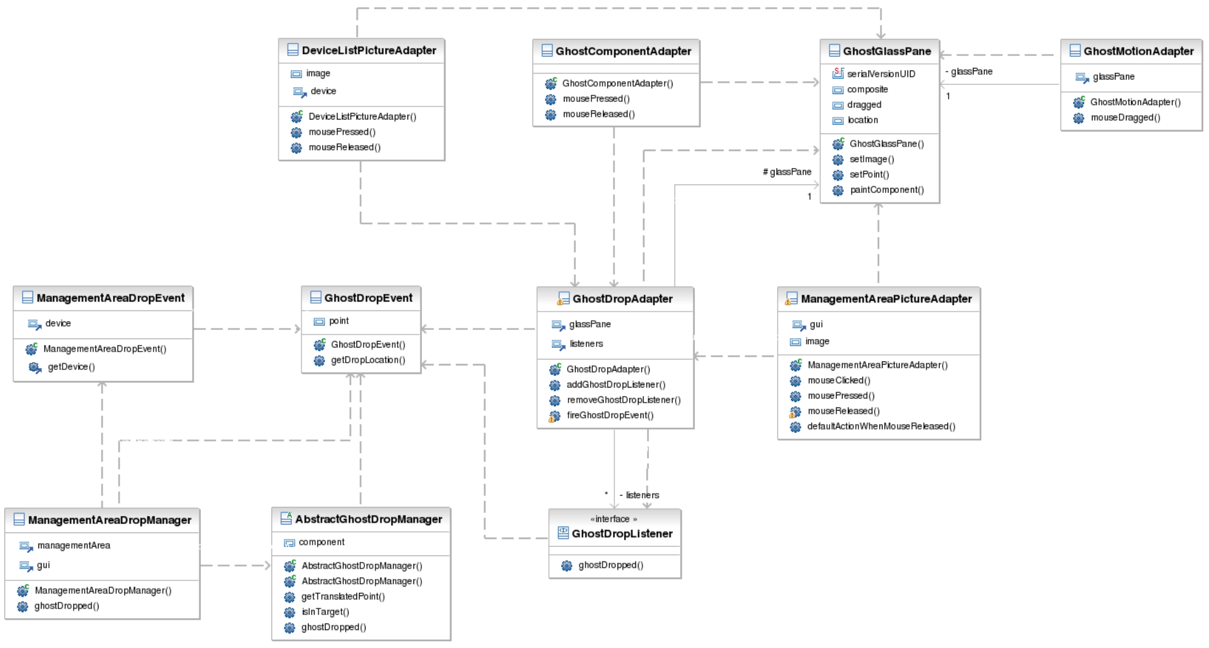
In order for our application to be *Drag & Drop* enabled, we had to create several classes that allowed us to do so, as java components are not *D&D* capable.

The main idea was to create a drag action handler, as well as a drop action handler, for the components of our application where any of these two actions could happen. In other words, the device list should handle both drag action (when a user wants to add a device to the management area) and drop action (when a user wants to remove a device from the management area).

The same concept applies to the management area : it has to handle drop action from a device which was dragged from the device list ; and it has to handle drag/drop action from itself.

We will start by explaining how the drop action is handled, either by the device list or the management area.

Figure 5.6 The Ghostpane class diagram



Class diagram for the ghostpane package.

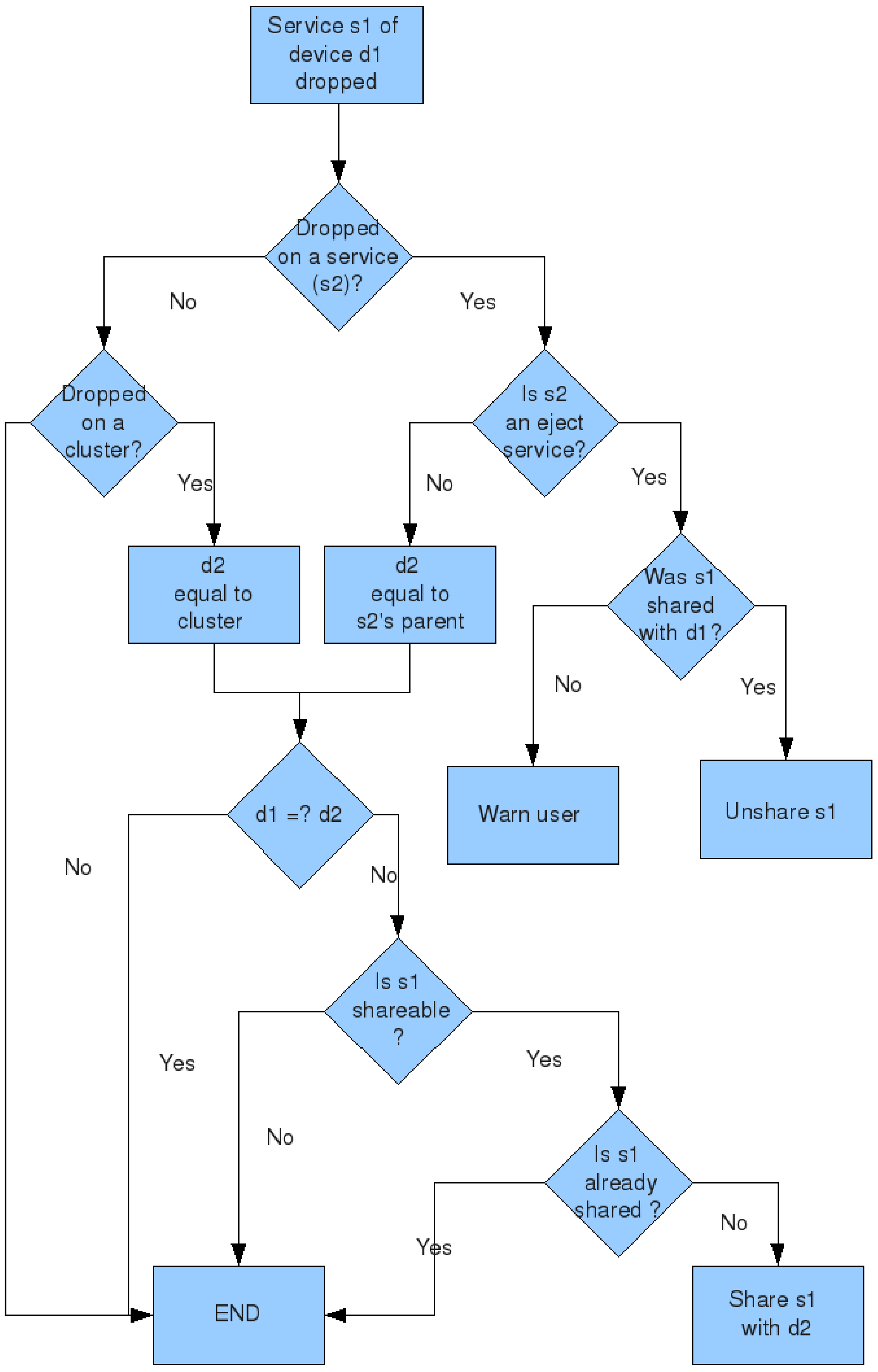
5.4.1 Drop action

Both the management area and device list extend the **JComponent** object. Basically what we are doing is adding some kind of listener on both component. These listeners, which we call drop managers, extend the **AbstractGhostDropManager** class, which itself implements the **GhostDropListener** interface. The **AbstractGhostDropManager** class specifies several methods that will be common for all the classes extending it: a constructor which takes a component in parameter (the component which will be 'listened' to), as well as a method whose duty is to check whether the drop action occurred on the listened component. Classes which will then extend the **AbstractGhostDropManager** class have to override a method called **ghostDropped**, which is a method that had to be implemented from the **GhostDropListener** interface. This method is the one called when a drop action occurs. Obviously, depending on the component it is listening to, the **ghostDropped** method will be different. For instance, this method for the management area first checks if the drop action occurred on it, and then adds the dragged device to the graph.

For the Management Area component, things are a little more complicated, because a drop action can occurs from within this component, not from another component like the device list. In other words, the management area has to be able to handle both a drag action on one of its elements (i.e. a device or a service), but more importantly to behave properly on a drop action.

Instead of describing it in words, the following flowchart diagram will summarize which conditions are checked when a drop is occurring.

Figure 5.7 Drop flowchart diagram

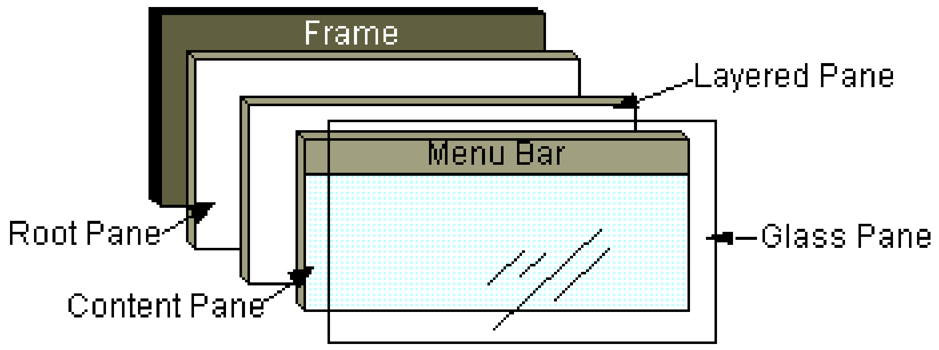


Flowchart diagram of the drop action

5.4.2 Drag action

To show the drag action being performed, we use a glass pane that overlays the application. It can be summarized as a layer on top of all the components constituting the graphical interface.

Figure 5.8 Integration of the glasspane on Swing components



[SUN - 2008] The Glasspane appears above all the other standard java components.

This allows us to display an icon which is related to the kind of object (device or service) being dragged.
Each class that handle a drag action extends the **GhostDropAdapter** class, which itself extends the **MouseAdapter** class. Hence, 2 methods are really critical: **mousePressed** and **mouseReleased**, the former being called when a user selects one object, the latter being called when the user finished his movement.
On the device list, the drag action is pretty simple : each label representing a device has a drag listener. The picture shown on the glass pane is the one representing the device.
This is different for the management area, mainly because there is only one drag listener on the whole component, not one for each device/service.

5.5 Virtual Keyboard

As it was described in the design chapter, we chose to implement a virtual keyboard for our application in order for the user to input text (for instance, when he has to log in).
In order to do this, we created a class called **VirtualKeyboard**, which instantiate a **JPanel** in which will be displayed the buttons constituting a regular keyboard. The configuration of the **VirtualKeyboard** relies on two configuration files:

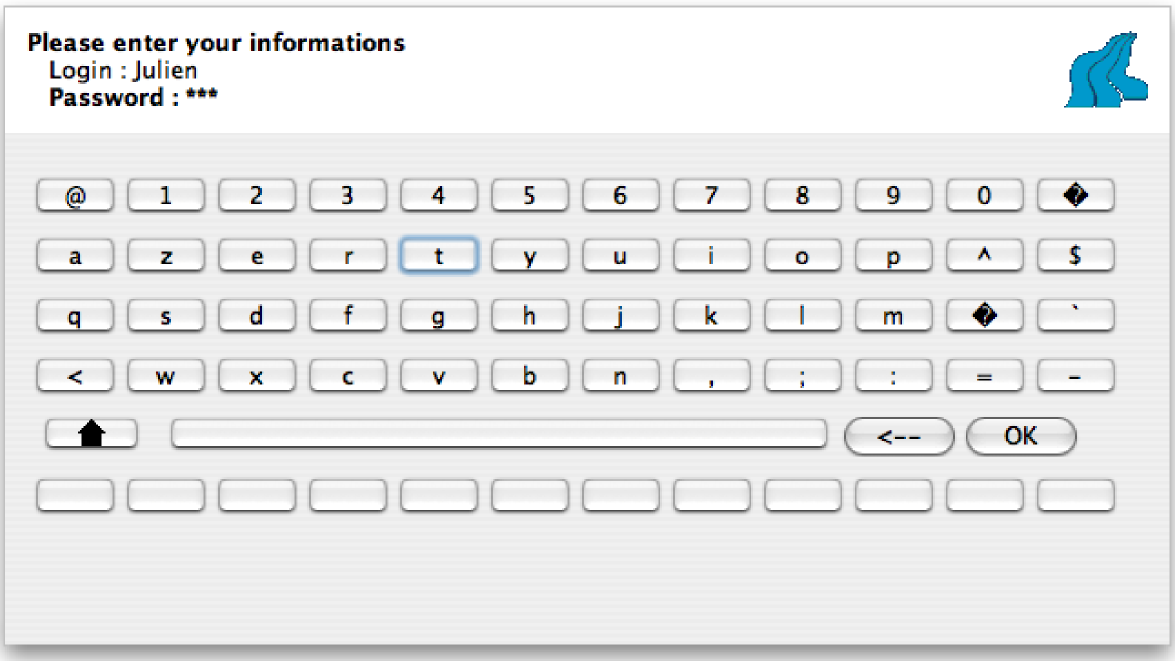
- **window.conf** specifying the position, width and height of the buttons
- **keyboard.conf** specifying which letters (upper case and lower case) should appear on the buttons

Using these configuration files means that it will be simple to modify the keyboard depending on the application (for instance, for changing the keyboard from an alphanumeric one to a numeric one).
The constructor of this class takes a special object in parameter : an object which implements the **VirtualKeyboardInputReceiver** interface. This interface specifies 3 methods:

- **addChar (String s):** this method is called each time a button is pressed on the keyboard. The parameter of this method is the letter corresponding to the pressed key.
- **removeChar():** this method is called when the backspace button of the virtual keyboard is pressed
- **returnKeyPressed():** this method is called when the Return button of the virtual keyboard is pressed.

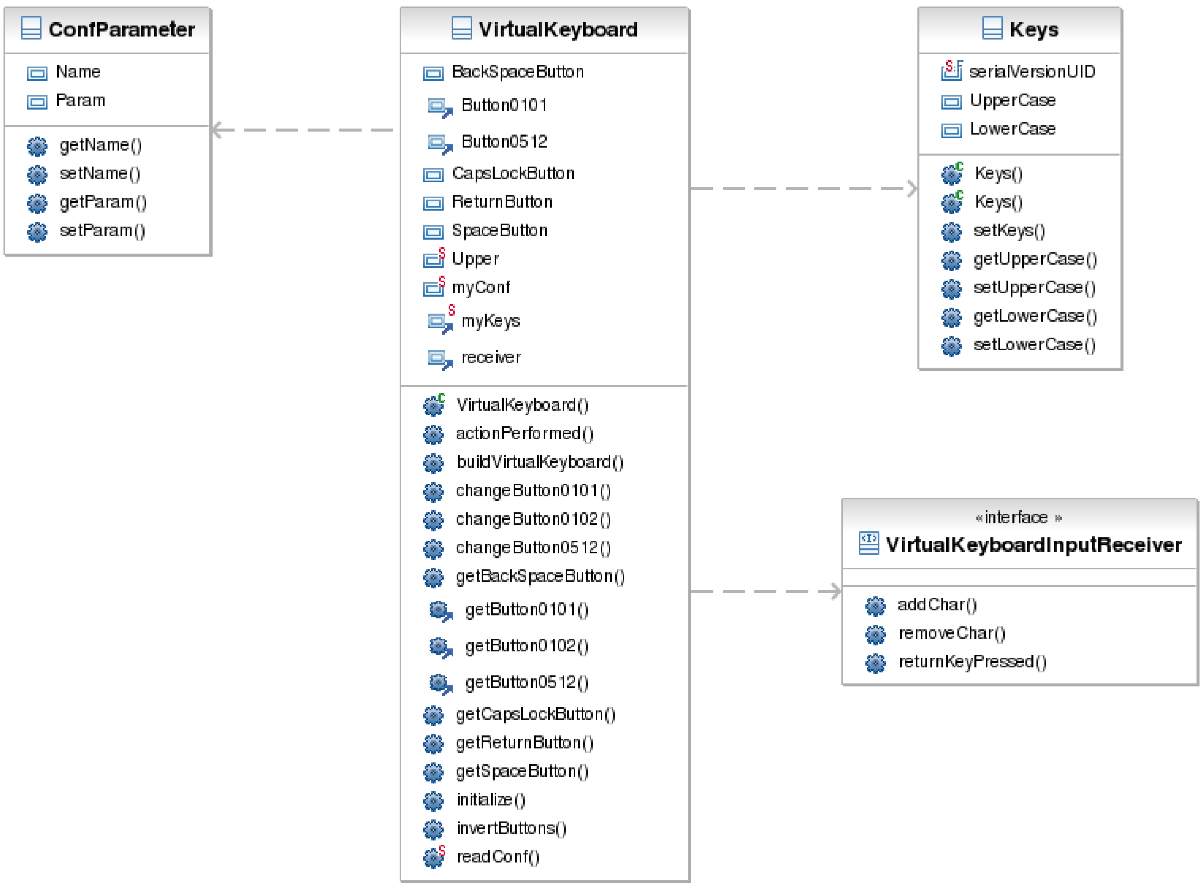
For our application, the only class implementing the **VirtualKeyboardInputReceiver** interface is the class **LoginPasswordPanel** from the GUI package.

Figure 5.9 Screenshot of our virtual keyboard implementation.



Design of our virtual keyboard.

Figure 5.10 The Virtual Keyboard class diagram



Class diagram for the virtualkeyboard package.

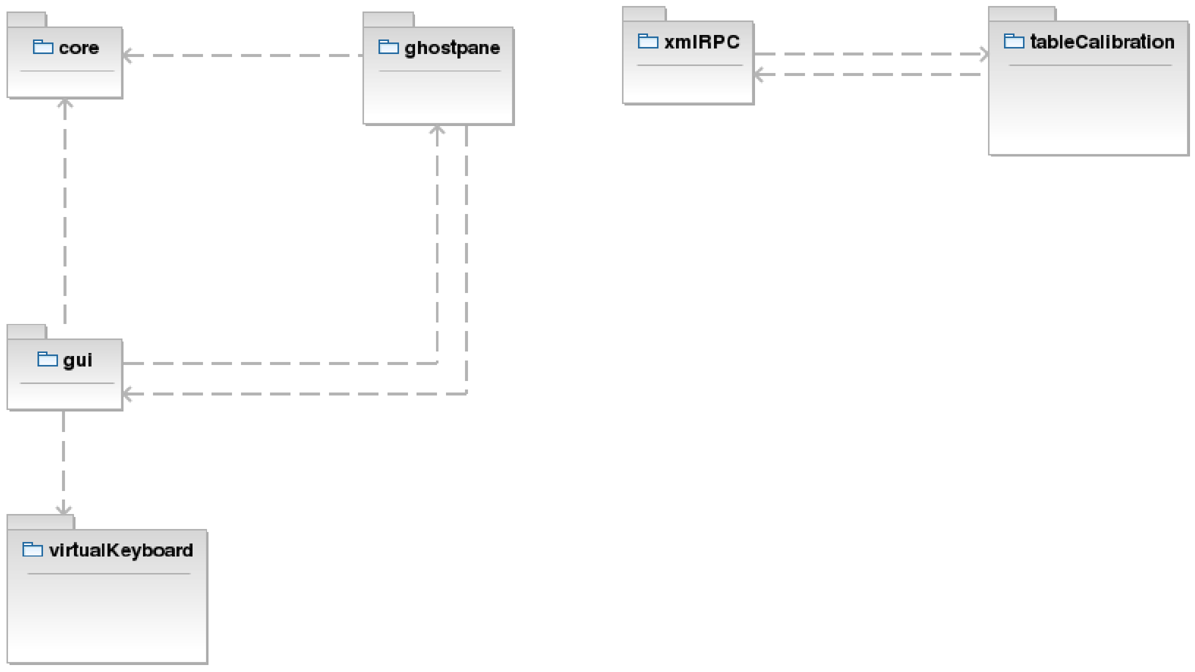
5.6 Magnet

Unfortunately, we were not able to implement anything regarding Magnet technologies, due to the lack of API. We were given source codes, but without any comments or how to make it work.

5.7 Implementation conclusion

In this section, we are only presnting the final class diagram showing relations between all the previous package discussed in the previous sections.

Figure 5.11 The general class diagram

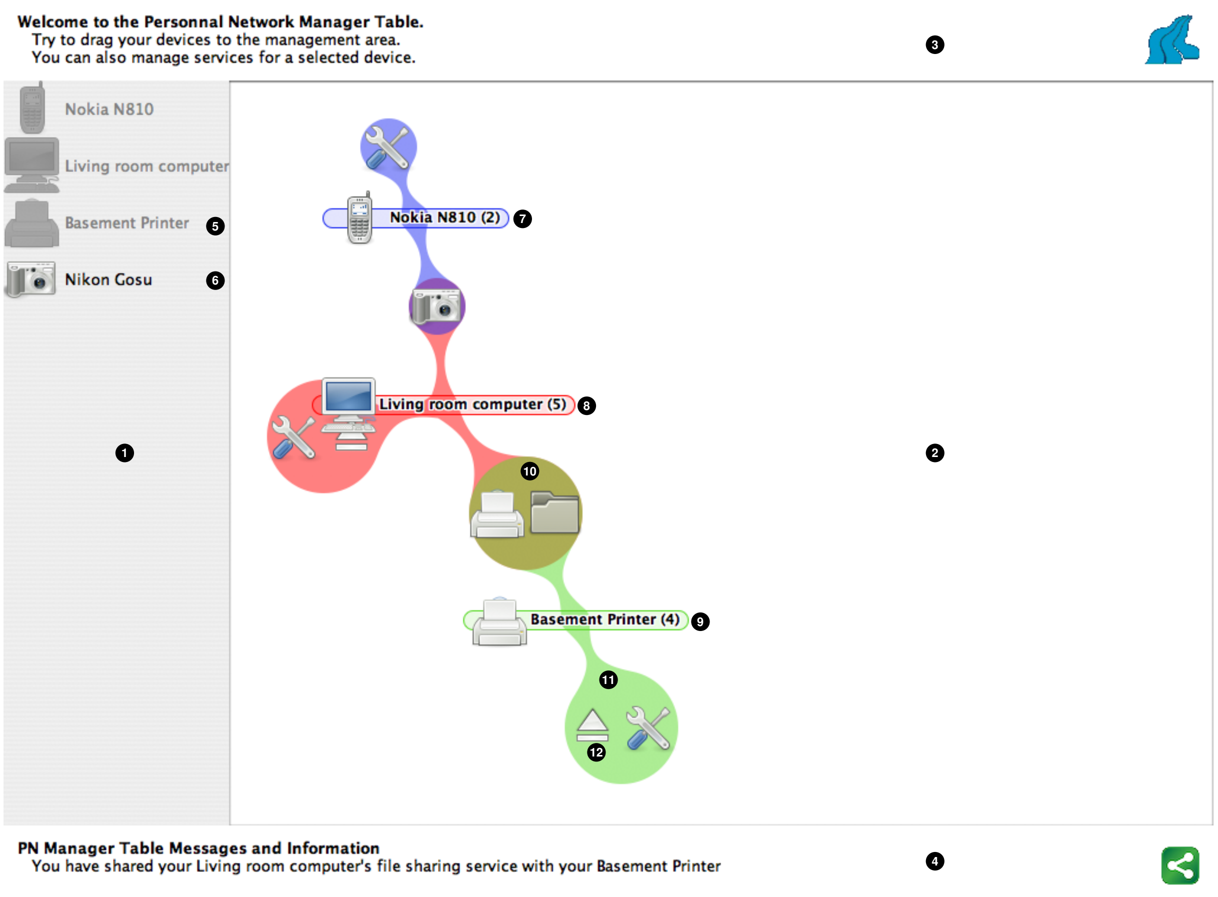


Overall class diagram showing relations existing between packages.

5.8 High-Fidelity prototype

In this section, we are going to show a screenshot of our GUI. This figure show the program used to realized tests discussed in the following chapter.

Figure 5.12 High-Fidelity prototype of the main screen.


Welcome to the Personnal Network Manager Table.
Try to drag your devices to the management area.
You can also manage services for a selected device.

PN Manager Table Messages and Information
You have shared your Living room computer's file sharing service with your Basement Printer

❶ The *List Area* is used to present to the user the list of the devices currently online within his PN, except for the devices which are displayed within the *Management Area*.

❷ The *Management Area* is the main area on the table. It is used to display device selected by the user.

❸, ❹ The *Top Area* is used to display a welcome message, and a short presentation of our software; whereas the *lower panel* displays tips or callback messages.

❺, ❻ It is really easy now to distinguish devices selected or not within the *List Area*. Effectively, the ones which are selected for a management purpose are grayish, and not dragable anymore, until the user remove them from the *Management Area*.

❼, ❽, ❾ The *Management Area* show currently three devices: the *Nokia N810*, the *Living Room Computer* and the *Basement Printer*

⓫, ❿ Colors are used to show separate *Clusters*. Each of them are linked to a device using colors. Moreover, when a *cluster* is shared, it color is a mix of both the colors of the sharing participants.

⓬ Icons are used to show services. This *eject button* icon is a little bit particular, because it only appears in a cluster when the devices owing the cluster is allowed to access to a resource. This icon is used to unshare this resources, by dropping the resource in it.

6 Tests

This chapter details the way application interface has been evaluated. Two parts have been tested: first, the user interface through a *usability tests*, then the functionality provided by our implementation through *functionality tests*.

For each of them, the testing procedure is first explained, then results are provided and commented.

6.1 Usability tests

The reason why we asked several persons to test our application was to identify any remaining bugs that we could have not found yet; as well as getting feedback from them concerning the general aspect of the application. Using their feedback, it was a lot easier for us to identify what was good and what was wrong.

Each test was put in the same situation : after explaining what lies behind the concept of personal network, we asked them to log on to the application (we created both a login and a password for each tester).

Once logged, they were asked to perform various actions, such as adding devices to the management area, share services between two or more device, un-share a service and remove the devices from the management area.

During the whole testing process, we asked them to say "out loud" what they were thinking and how they thought the application would react to their action. This way, we were able to obtain another level of criticism toward the application.

After all the tasks we asked them to perform were done, we gave them a user questionnaire in which they were able to "grade" our application based on different criterions. On top of that, should they wanted to, they were able to leave some more comments.

6.1.1 Test process

The test were performed using the table top display of the multi-touch table. At that time, there were still issues with the accuracy of the coordinates. In order to bypass this issue, we handled all the input ourselves using the mouse. Even if this is not totally transparent to the tester, we thought doing the tests this way would be more relevant than having the application perform actions that were actually not asked by the testers. Due to a lack of time, we were not able to perform another series of test.

The tests have been performed with four people who were asked to perform the actions that can be found in appendix.

6.1.2 Task evaluation

The aspects that were observed during those tests were the user's behavior while using the application, and his actions in order to interact with it, successfully or not.

6.1.3 Live feedback

As it has been explained in the first paragraph of this chapter, we asked the testers to say what they were thinking. We were then able to get reactions we would not have had if based solely on the comments written on the grading sheet.

One of the tester found the concept of 'Management Area' not being clear. Even if, on the upper panel of the application, there is information about how to make it work, it made us realize that we should emphasize it. The same tester found the 'ok' button on the virtual keyboard to be disturbing due to the fact that when a login has been entered, the user would rather have a 'Next' button than an 'ok' button. Two other testers had another kind of issue with the virtual keyboard, where they kept the shift button pushed.

We also find out that overall, when the user reads what is displayed on the screen (on both lower and upper panels), he/she tends to use the application the way it is supposed to. For instance, one of the tester did not read the information, and as a matter of fact made the mistake of simply "clicking" on a device to add it to the management area, instead of dragging it and dropping it.

Most of the users found the icons used as being representative enough, but the idea of having a tool tip box appearing when selecting a device was a common request.

6.1.4 Results

Figure 6.1 Diagram 1: ease of use of our application

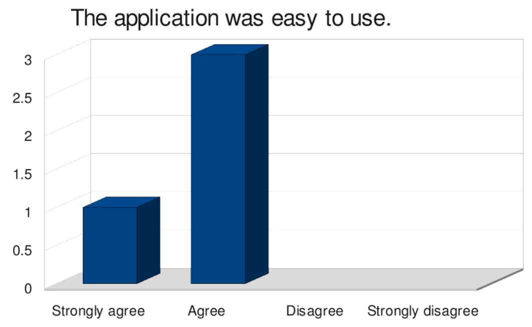


Chart Diagram showing the answers to the ease of use of our application.

As we can see on this diagram, the majority of our test-users were satisfyed by the usability of the application. They found the application to be easy to use.

Figure 6.2 Diagram 2: pleasantness of our application

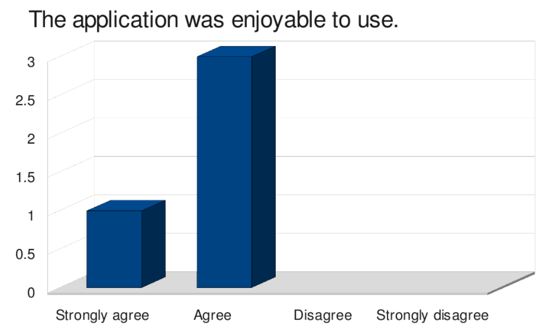


Chart Diagram showing the answers regarding the pleasantness of our application.

This diagram shows that besides our application is easy to use, it remains enjoyable to use.

Figure 6.3 Diagram 3: touch-screen functionning

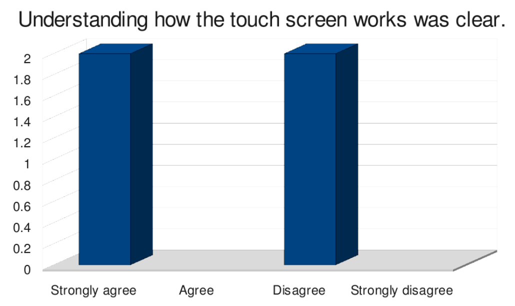


Chart Diagram showing the answers to the easiness of use of the touch-screen.

The results shown in this graph can be explain by the fact that some users are more at ease with uncommon input devices than others. These users are generally the one who are not at ease with mobile devices.

Figure 6.4 Diagram 4: functionning of our program

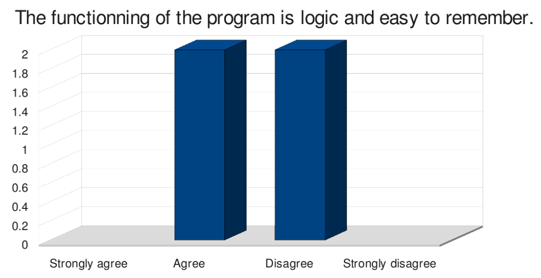


Chart Diagram showing the answers to the question regarding the users perception of the functionning of our application.

The mitigated results of this graph was an indication of a necessity for improvements. From the live comments we get, the improvments regard partly the virtual keyboard interface, and partly the help panel visibility.

Figure 6.5 Diagram 5: Design of our application

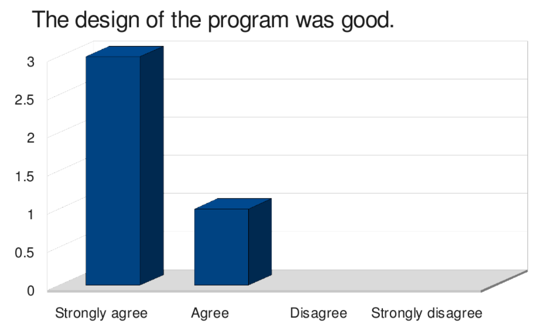


Chart Diagram showing the answers to the design of our application.

Thanks to the Aduna API, our application offered a really good graphical interface design which was praised by all the test users.

Figure 6.6 Diagram 6: meaning of the icon understandable

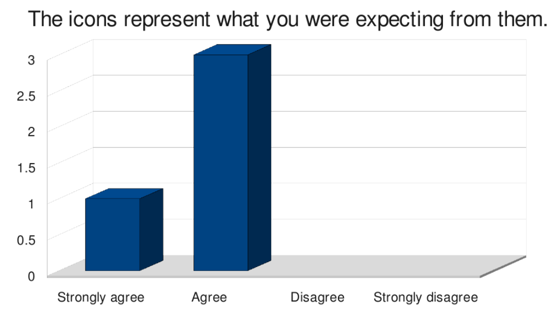


Chart Diagram showing the answers to the understandability of the icons describing both the devices and their services.

Despite previous concern regarding the meaning of some icons, the users found that a given icon represent what he/she expected it to be.

Figure 6.7 Diagram 7: reliability of our application

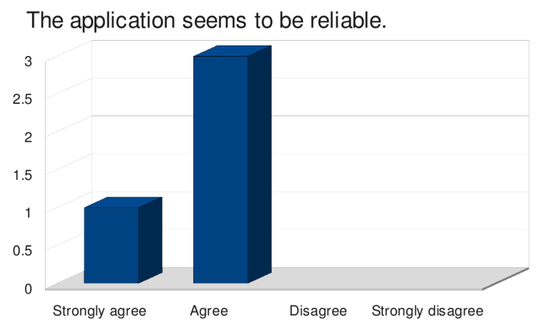


Chart Diagram showing the answers to the reliability of our application

Not a single bug was found during the usability tests, hence the results shown by this graph.

6.2 Functionality tests

Regarding the functionality tests we done, these ones only were performed to compare our work to the previsions made at the beginning of out project. To achieve such a comparison, we compared the High-Fidelity prototype to the use-cases described in Table 3.1,Table 3.2 , Table 3.3 , Table 3.4 , and in Table 3.5 .

As a reminder, the basic scenario defined in Section 3.3 was to allow an end-user to share easily data stored on a mobile phone (for instance, pictures taken with the embedded camera, songs stored in the memory card, phone numbers of his contacts, . . .) between this cellular and a computer, both of them being part of his PN.

At the time we are finishing our report, such a scenario is globally feasible. From a user point of view, we are able to display in a friendly way a PN. After the user logged in thanks to our virtual keyboard, he access the management part of our application. In this application, devices are ordered in a list, and the user can select some of these online devices to manage their sharing relationships. Once a device is selected, it appears in the *Management area*. This area allows him to manage resources and services in a more simple way than with the *pnmanager*. Effectively, to share (or unshare) something, he only needs to drag and drop the icon representing the service to the recipient device. To unshare such a resource, he can still do it with the drag and drop paradigm we introduced by dropping the shared service on the eject icon. Finally, the end-user can remove a device from the *Management area* by dragging and dropping it back in the *List area*.

Among the features we did not get enough time to implement is the multi-touch support of our application. Indeed, this multi-touch support is not handled by neither Java nor windows; whereas the Table is capable of detecting several fingers on top of its screen. However, we tried to simplify the user interface to the max, and we are thinking that having only one paradigm as the *Drag & Drop* one is enough for the end-user. More than one would confuse the user.

7 Conclusion and perspective

7.1 The project

As already stated in the Chapter 1, this project aimed to bring to the end-user a new way to manage the sharing of services among his PN devices. We found that providing him a new kind of interface such as a multi-touch tabletop display could be enjoyable. To reach this goal, we did not start from scratch. We took advantage of the demeTouch project lead in 2007 by Alexandre Fleury. This one already built a multi-touch tabletop display.

To achieve the goal of bringing a user-friendly application allowing the end-user to manage his PN, we divided the project into several part: a design part and an implementation one.

We began by designing a low-fidelity prototype and the whole system to make it able to take advantage of such a display. We tried to get feedbacks about this design. Then, we developed a high-fidelity java prototype. This second step finished by usability-tests which were really interesting regarding the new feedbacks we get from test users.

7.2 Project achievement

We are glad for having build such a system. As described in the last subsection, we have written a software application able to take advantage of the touch display, but as this one is relying on the Aduna Clustermap API, we did not succeed to benefit from the multi-touch capabilities of the demeTouch fingerDetector. On the other hand, our software is able to display and manage in a really easy way PN devices, and that was the main goal we tried to set has been reached in this project.

But the result of the project is far from perfect. We are not able to bring to the user a really new kind of interface as the Surface© table or the iPhone© brings. Moreover, we have not been able to take advantage of the Magnet project, because of its lack of API. However, when these API will be defined, it should be really easy to interface our application with Magnet.

As a conclusion, we can answer the three question we tried to answer all along this report and this project. These questions were:

1. *What would be the best user-friendly way to represent devices and services within a Personal Network?*

   For this first question, using a clustered map as we have done, coupled with a simplifyed graphical interface let us design an application which was defined as really user-friendly by our test-users.

2. *Is it possible to improve the already existing Magnet command line interface?*

   For this second question, we can answer yes, because the only tool which is currently availlable to the developpers is a command line interface. There is no tool designed for end-users.

3. *How can we benefit from the use of a multi-touch table?*

For this last question, we cannot really answer, because we did not succesfully manage to use the multi-touch capability of the **fingerServer**. However, we think the idea of providing to the end-user tabletop display in public area such as Internet Café or library could really be welcomed by users when the Magnet PN project will be a more widely spread technology.

7.3 Further development

There are many possible improvements left that could be brought to the system. Work could be done on these remaining points: the table-software interface, the multi-touch capabilities.

Table-software interface Currently, the XML-RPC communication between these two module works perfectly. The main annoyance we encounteredf comes from the fingerDetector latency. This latency can be reduced by optimizing the algorithm parameters.

Multi-touch capabilities As our high-fidelity is based on the Aduna Clustermap API, we can not bring to the end-user a multi-touch system, even if the fingerDetector is able to detect several finger on top of the screen. Bringing multitouch capabilities to our software require to re-develop more than just a cluster map API. Currently Java does not provide support for multiple input devices. Thus such a capability must be handled by software developers from scratch. Bringing such a multi-touch support into java could be a project by itself.

Graphical User Interface Finally, the application needs to be really easy to use. Thus, another improvement could be to enhance the graphical interface by redesigning icons, and classification of devices and services within the Clustermap API.

The combination of these three points would improve greatly the system. Bringing to the end-user the multi-touch support would be an interesting starting point for further projects.

7.4 Personal achievement

The work done during this project has been rewarding for several reasons.

We already had some backgrounds corresponding to this project such as java programming knowledge or some other skills about GUI. But this project has been mainly focused on the usability part. This was really interesting to mainly emphasize on the user interaction with our system because we are not used to such work. Moreover, even if we do not succeed to bring a multi-touch application to the end-user, it was really interesting to have a look at what is currently achieve within other projects. The Surface© table or the iPhone© introduce new user-interaction paradigm which will quickly spread.

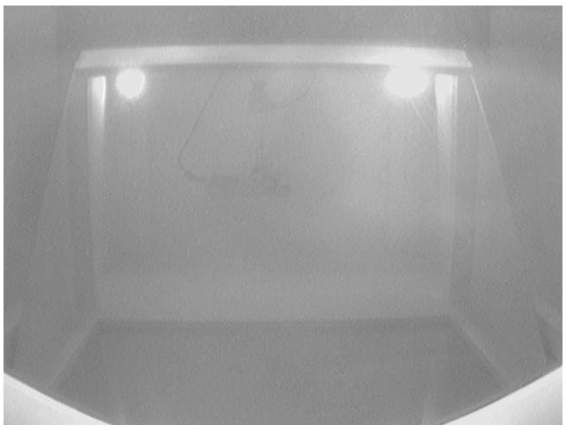
To conclude, we would like to say that we were glad to work on that project which was dealing with so many different matters going from the Affine transform mathematical tool to the Magnet project, and with whom we had got many new skills.

A

demeTouch software

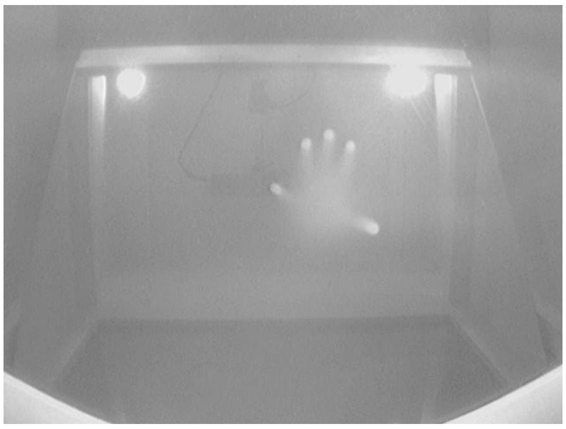
A.1 demeTouch image processor explained

Figure A.1 Initial Step



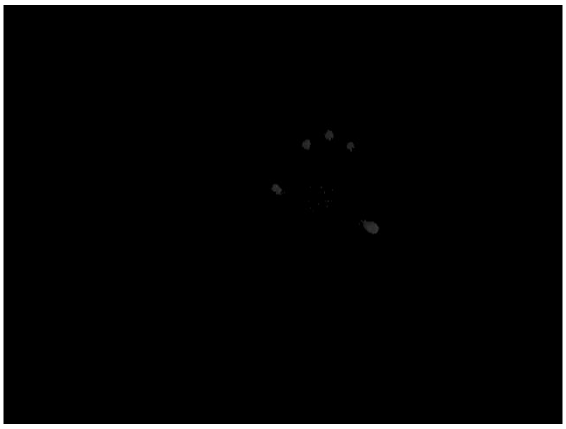
For the purpose of finger detection, the reference image should correspond to the scene without fingers laying on the table.

Figure A.2 Step 0: Retrieval of the current image



The following algorithms rely on this captured image. Basically the main ones are the subtraction and the binarization. Thanks to these ones, Alexandre is able to clearly show elements differing from one reference picture R to the captured one.
Fingers are more or less visible depending on the infrared beams repartition on the table's surface. The more infrared light, the more reflection from the fingers, and as a consequence the more visible. Unfortunately, infrared beams are not equally spread on the table's surface, making the fingers less reflective in some areas.

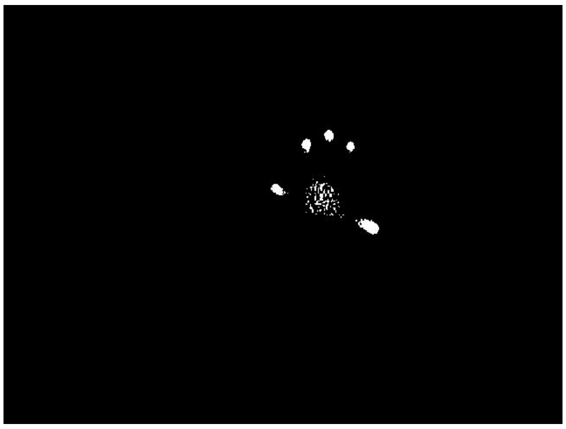
Figure A.3 Step 1: Subtraction



This operation processes a pixel-by-pixel subtraction between the current image A and a reference image R .
For the purpose of finger detection, the reference image should correspond to the scene without fingers laying on the table. In that case, the ideal resulting image would have most pixels set to 0 (as most of the image does not change); only the zones where fingers appear being set to a non-zero value. In reality, due to the noise introduced by the camera, the current image is never perfectly equal to the reference one, even in zones without fingers.
To handle this problem, a threshold ε_S (ranged from 1-255) is introduced so that if the subtraction result is less than ε_S , the two images are assumed equal.

$$i_{(x,y)} = \begin{cases} \mid A_{(x,y)} - R_{(x,y)} \mid & \text{if} \mid A_{(x,y)} - R_{(x,y)} \mid > \varepsilon_S \\ 0 & \text{otherwise} \end{cases} \tag{A.0}$$

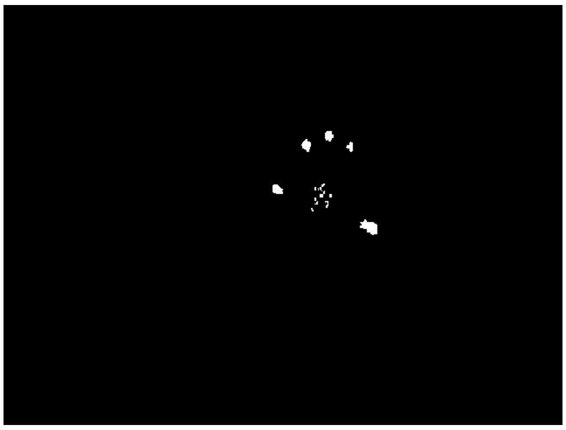
Figure A.4 Step 2: Binarization



Binarizing an image consists in setting its pixel values either to 0 (black) or 255 (white).
In the present situation, to remove the specular reflections of the two infrared lights that appear on the camera, and which are characterized by being extremely bright (in these areas, pixel values are very close to 255), Alexandre uses two thresholds ε_{Bi_d} (threshold down) and ε_{Bi_u} (threshold up), so that only pixels with a value comprised in between these thresholds are set to 255.

$$i_{(x,y)} = \begin{cases} 255 & \text{if } \varepsilon_{Bi_d} < A_{(x,y)} < \varepsilon_{Bi_u} \\ 0 & \text{otherwise} \end{cases} \quad \text{(A.0)}$$

Figure A.5 Step 3: Holes Removal

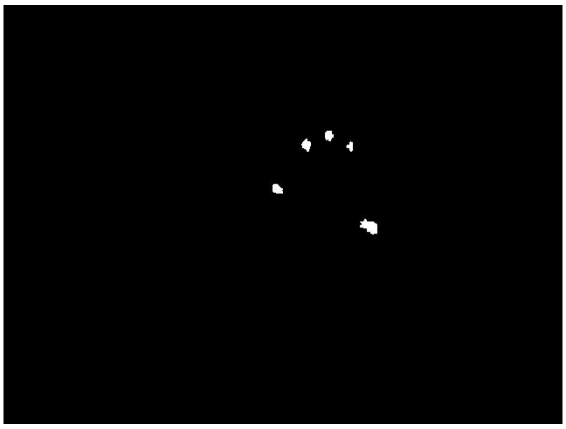


This technique is used to make the finger representation a plain continuous area instead of a discontinuous set of spots.
After the binarization step, Alexandre processes to the Holes Removal step. He firstly defines windows (6x6) inside which pixel values are counted (values being either 0 or 255 after binarization). Then, all pixels in the windows are set to the most recurrent value.

$$i_{(x,y)} = \phi^k \cdot i_{(x,y)}, \text{ where } \phi^k = \frac{255}{\max i^k_{(x,y)}} \quad \text{(A.0)}$$

After the previous methods have been applied, the original image is mostly composed of black pixels, only the areas showing potential fingers being white. From this image is extracted the location of these potential fingers. At the end of the process, each spot s is defined as a collection of white consecutive pixels.

Figure A.6 Step 4: Size Discard

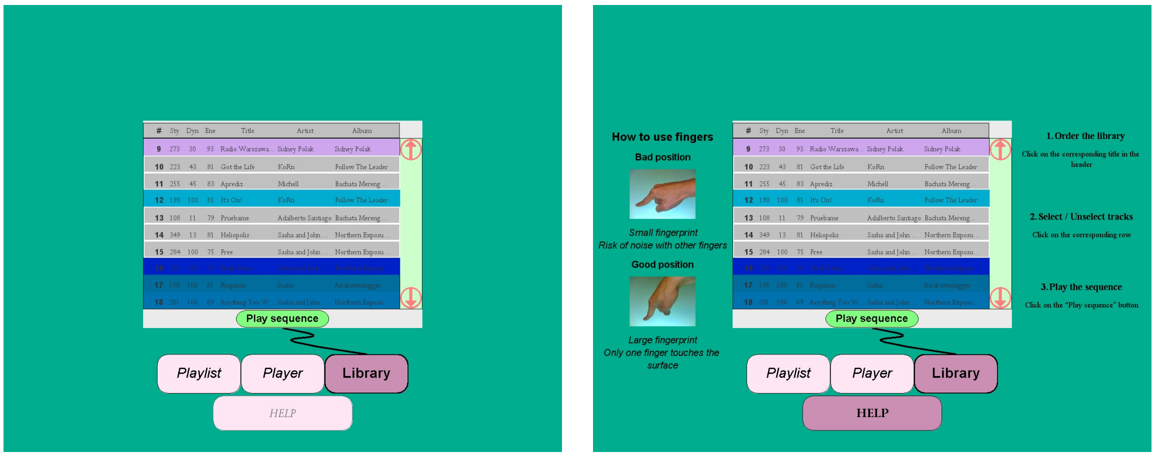


After having detected spots, the next step is to discard the ones which has either an irregular shape or a size that cannot correspond to a finger.
The set of spots S introduced during the spot creation's explanation is now regulated by two thresholds ε_d (threshold down) and ε_u (threshold up) controlling the spots' size.

$$S = \sum_{k=0}^{K} \sum_{n=0}^{N_k} i_n = 255 \text{ if } \varepsilon_d < N_k < \varepsilon_u \qquad \text{(A.0)}$$

A.2 demeTouch UI

Figure A.7 The Library screen



#	Sty	Dyn	Ene	Title	Artist	Album
9	273	30	93	Radio Warszawa…	Sidney Polak	Sidney Polak
10	223	43	81	Got the Life	KoRn	Follow The Leader
11	255	45	83	Aprediz	Michell	Bachata Mereng…
12	190	100	81	It's On!	KoRn	Follow The Leader
13	108	11	79	Pruebame	Adalberto Santiago	Bachata Mereng…
14	349	13	81	Heliopolis	Sasha and John …	Northern Exposu…
15	284	100	75	Free	Sasha and John …	Northern Exposu…
16	230	100	78	Rock Stone	Sasha and John …	Northern Exposu…
17	198	100	61	Requiem	Sasha	Airdrawndagger
18	201	100	69	Anything You W…	Sasha and John …	Northern Exposu…

How to use fingers

Bad position

Small fingerprint
Risk of noise with other fingers

Good position

Large fingerprint
Only one finger touches the surface

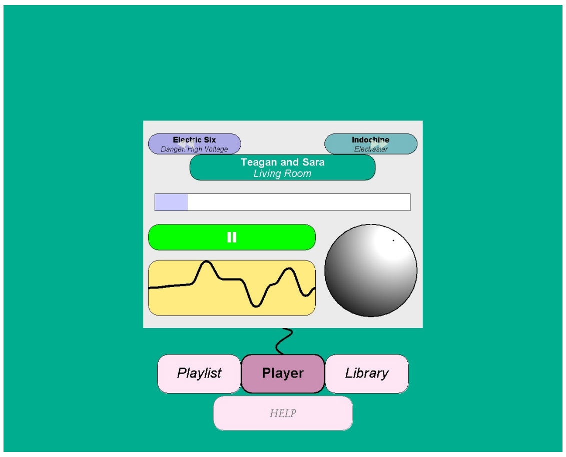
1. Order the library
Click on the corresponding title in the header

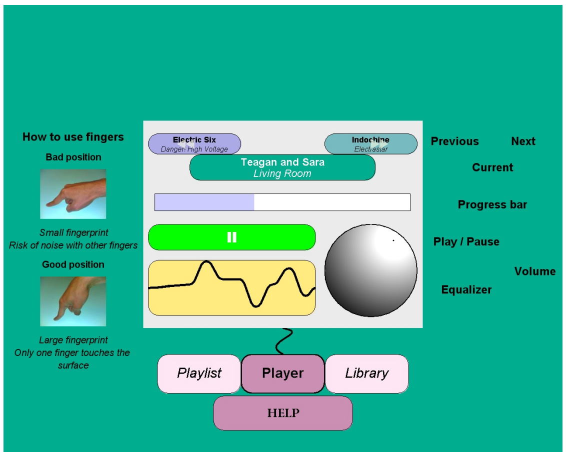
2. Select / Unselect tracks
Click on the corresponding row

3. Play the sequence
Click on the "Play sequence" button

The Library screen: This screen shows the library and the necessary buttons to let the user browses it. If the user feels it needs help, he can display it around the playlist visual elements thanks to a button.

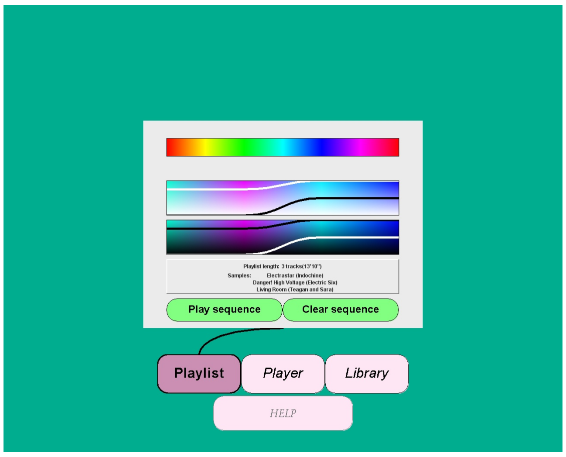
Figure A.8 The Player screen

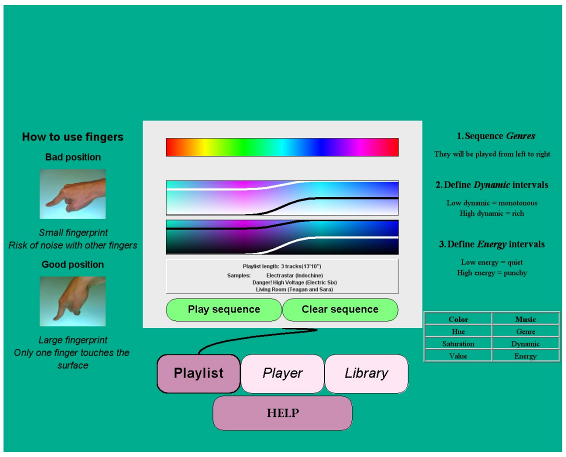




The Player screen: This screen shows the player and the necessary buttons to let the user listen to his music. If the user feels it needs help, he can display it around the player visual elements thanks to a button.

Figure A.9 The Playlist screen


Playlist length: 3 tracks(13'10")
Samples: Electrastar (Indochine)
Danger! High Voltage (Electric Six)
Living Room (Teagan and Sara)


Playlist length: 3 tracks(13'10")
Samples: Electrastar (Indochine)
Danger! High Voltage (Electric Six)
Living Room (Teagan and Sara)

1. Sequence *Genres*
They will be played from left to right

2. Define *Dynamic* intervals
Low dynamic = monotonous
High dynamic = rich

3. Define *Energy* intervals
Low energy = quiet
High energy = punchy

Color	Music
Hue	Genre
Saturation	Dynamic
Value	Energy

The Playlist screen: This screen shows the playlist editor and the necessary buttons to let the user use it. If the user feels it needs help, he can display it around the editor visual elements thanks to a button.

B Abbreviations

API	Application Programming Interface
ARP	Address Resolution Protocol
BT	Bluetooth
CASD	Context Aware Service discovery
CD	Context Discovery
CM	Context Manager
CMF	Context Management Framework
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
ER	Edge Router
FTP	File Transfer Protocol
GSM	Global System for Mobile Communication
GUI	Graphical User Interface
GW	Gateway
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
IPSec	Internet Protocol Security
LAN	Local Area Network
LLC	Logical Link Control
MAC	Medium Access Control
MAGNET	My Adaptive Global Net
MSDP	MAGNET Service Discovery Platform
MSMP	MAGNET Service Management Platform
NAT	Network Address Translation
P2P	Peer to Peer
PAN	Personal Area Network
PDA	Personal Digital Assistant
PDP	Peer Discovery Protocol
PHY	Physical Layer
PN	Personal Network
PN-F	Personal Network Federation
PNDS	Personal Network Directory Service
P-PAN	Private Personal Area Network
OOP	Object Oriented Programming
OS	Operating System
QoS	Quality of Service
R&SD	Resource and Service Discovery

RPC	Remote Procedure Call
SA	Service Agent
SCMF	Secure Context Management Framework
SD	Service Discovery
SGN	Service Gateway Node
SHA	Secure Hash Algorithm
SMN	Service Management Node
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SSDP	Simple Service Discovery Protocol
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UCL	Universal Convergence Layer
UDP	User Datagram Protocol
UI	User Interface
UPnP	Universal Plug ’n’ Play
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WAN	Wide Area Network
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network
WSDL	Web Services Description Language
WWW	World Wide Web
XML	eXtensible Markup Language

C Glossary

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A

Application Programming Interface (API)

An API is a source code interface that an operating system or library provides to support requests for services to be made of it by computer programs. The concept is meant to represent any well defined interface between two separate programs.

http://en.wikipedia.org/wiki/API

D

Drag & Drop (D&D)

In computer graphical user interfaces, drag-and-drop is the action of (or support for the action of) clicking on a virtual object and dragging it to a different location or onto another virtual object. In general, it can be used to invoke many kinds of actions, or create various types of associations between two abstract objects.

As a feature, support for drag-and-drop is not found in all software, though it is sometimes a fast and easy-to-learn technique for users to perform tasks.

The basic sequence involved in drag-and-drop is:

- Press, and hold down, the button on the mouse or other pointing device, to "grab" the object,
- "Drag" the object/cursor/pointing device to the desired location,
- "Drop" the object by releasing the button.

http://en.wikipedia.org/wiki/Drag_and_drop

M

My personnal Adaptative NETwork (MAGNET)

MAGNET Beyond is a worldwide R&D project within Mobile and Wireless Systems and Platforms Beyond 3G. MAGNET Beyond will introduce new technologies, systems, and applications that are at the same time user-centric and secure. MAGNET Beyond will develop user-centric business model concepts for secure Personal Networks in multi-network, multi-device, and multi-user environments.

MAGNET Beyond has 30 partners from 15 countries, among these Industrial Partners, Universities, and Research Centres. MAGNET Beyond is a continuation of the MAGNET project.

MAGNET goals is to define Personnal Network by develloping and enhancing actual networking solution, both at a physical level (enhancing the bandwith, range and other physical metrics of devices' antenas for instance) and a an application level (creation of communication protocols and solution ensuring security while communicating with other PN devices). Finally, this project also work on the definition of use-case scenarios which would be feasible thanks to such a technology.

http://www.ist-magnet.org/

P

Personal Network (`PN`)

In Magnet, a Personal Network is an extension of the well known PAN (Private Area Network). In fact, a PN is composed of several PAN located in several places. For instance, a PN can be composed of devices (computer for instance, but also all the other networking devices) located in the profesional PAN of a user. This PN can also contain devices belonging to the Home PAN. In the Magnet terminology, such a PAN is called a "cluster". A PN is an virtual network overlaying the public (or even private) infrasctructure it uses to communicate. This virtual private network (VPN) is used in combination with encryption keys to ensure both, privacy, security, anonanymity regarding the external networks.

Magnet goal is to build network from a user-centric point of view. This goal is ensured by the notion of service provider. That is to say that devices connected to a PN can benefit from the other devices services they offer to the network.

R

Robot

The Robot java class is used to generate native system input events for the purposes of test automation, self-running demos, and other applications where control of the mouse and keyboard is needed. The primary purpose of Robot is to facilitate automated testing of Java platform implementations.

Using the class to generate input events differs from posting events to the AWT event queue or AWT components in that the events are generated in the platform's native input queue. For example, Robot.mouseMove will actually move the mouse cursor instead of just generating mouse move events.

Note that some platforms require special privileges or extensions to access low-level input control. If the current platform configuration does not allow input control, an AWTException will be thrown when trying to construct Robot objects. For example, X-Window systems will throw the exception if the XTEST 2.2 standard extension is not supported (or not enabled) by the X server.

http://java.sun.com/j2se/1.5.0/docs/api/java/awt/Robot.html

X

XML-RPC (`XML-RPC`)

XML-RPC is very simple Remote Procedure Call protocol created by Dave Winer of UserLand Software in 1998 with Microsoft. It uses XML to encode its calls and HTTP as a transport mechanism. Its entire description can be printed on two pages of paper.

Example of the protocol requests and returns can be found on the wikipedia article located at http://en.wikipedia.org/wiki/XML-RPC

D Bibliography

D.1 Books & Reports

[Fleury - 2007] Alexandre Fleury, *demeTouch*, a New Approach to Human Music Interaction, Copyright © 2007 Alexandre Fleury, Aalborg University, http://projekter.aau.dk/projekter/-research/demetouch (9931666)/ (05/22/2007) .

[Fluit et al - 2005] Christiaan Fluit, Marta Sabou, and Frank van Harmelen, *Ontology-based Information Visualisation: Towards Semantic Web Applications*, Copyright © 2005 Vladimir Geroimenko, Springer Verlag.

[Larsen - 2005] Søren Larsen, *Puka Finder: An Introduction to Tabletop Displays*, Copyright © 2005 Søren Larsen, Aalborg University.

[Spence - 2007] Robert Spence, *Information Visualization*, Design for Interaction, Copyright © 2007 Pearson Education Limited, 0-132-0655-09, Pearson Prentice Hall, http://www.pearsoned.co.uk/spence (05/22/2007) .

D.2 Magnet related public documentation

[Balken et al - 2006] R. Balken, J. Haukrogh, J. L. Jensen, M. L. Jensen, L. J. Roost, P. N. Toft, R. L. Olsen, and H.-P Schwefel, *Wireless Personal Communications*, Context-Sensitive Service Discovery Experimental Prototype & Evaluation, Copyright © 2006 Springer, 40, 2007.

[Bauer et al] M. Bauer, R.L. Olsen, M. Jacobssen, M.L. Sanchez, J. Lanza, M. Imine, and N. Prasad, *Context Management Framework for MAGNET Beyond*

[Ghader et al] Majid Ghader, Rasmus L. Olsen, Venkatesha Prasad, Martin Jacobsson, Luis Sanchez, Jorge Lanza, Wassef Louati, Marc Girod Genet, Djamal Zeghlache, and Rahim Tafazolli, *Service Discovery in Personal Networks; design, implementation and analysis*

[Ghader et al - 2004] Majid Ghader, Rasmus Olsen, Luis Sanchez, Jorge Lanza, Luis Muñoz, Jeroen Hoebeke, Ingrid Moerman, Marina Petrova, Matthias Wellens, Marc Girod-Genet, Wassef Louati, Khaled Masmoudi, Djamal Zeghlache, Nina Köstering, Sami Lehtonen, and Rahim Tafazolli, *Resource and Service Discovery: PN Solutions*, Copyright © 2004 Magnet Beyond, Magnet Ref: D2.1.1, http://www.ist-magnet.org/-GetAsset.action?contentId=939330&assetId=939353 (05/22/2007) .

[Ghader et al - a] Majid Ghader, Jeroen Hoebeke, Gerry Holderbeke, Ingrid Moerman, Martin Jacobsson, Venkatesha Prasad, N. I. Cempaka Wangi, Ignas Niemegeers, and Sonia Heemstra De Groot, *Personal Network Federations*

[Hoebeke et al] Jeroen Hoebeke, Gerry Holderbeke, Ingrid Moerman, Wajdi Louati, Marc Girod Genet, Djamal Zeghlache, Luis Sanchez, Jorge Lanza, Mikko Alutoin, Kimmo Ahola, Sami Lehtonen, and Jordi Jaen Pallares, *Personal Networks: From concept to a demonstrator*

[Magnet - 2008] Magnet Project, *MAGNET Beyond: Making Personal Networks happen!*, Copyright © 2006-2008 Magnet Beyond, Magnet Beyond, http://www.ist-magnet.org/-GetAsset.action?contentId=783096&assetId=783222 (05/22/2007) .

[Petrova et al - 2005] Marina Petrova, Matthias Wellens, Simon Oosthoek, Djamal Zeghlache, Wajdi Louati, Wassef Louati, Marc Girot-Genet, Khaled Masmoudi, Luis Sanchez, Jorge Lanza, Luis Muñoz, Rasmus Olsen, Homare Murakami, Dimitris Kyriazanos, Michael Argyropoulos, John Floroiu, Yacine Rebahi, Shahab Mirzadeh, Martin Jacobsson, and Venkatesha Prasad, *Overall secure PN architecture*, Copyright © 2005 Magnet Beyond, Magnet Ref: D2.1.2/D4.1.3, http://www.ist-magnet.org/-GetAsset.action?contentId=939330&assetId=939345 (05/22/2007) .

[Sivarajah et al - 2005] Kumarendra Sivarajah, Thafer Sulaiman, Hamed Al-Raweshidy, Terence Dodgson, Merkouris Karaliopoulos, and Majid Ghader, *Specification of PN Networking and Security Components*, Copyright © 2005 Magnet Beyond, Magnet Ref: D2.3.1, http://www.ist-magnet.org/GetAsset.action?contentId=939330&assetId=939346 (05/22/2007) .

[WP1 - 2006] WP1, *Users, Pilot Services and Market*, Copyright © 2006 Magnet Beyond, http://www.ist-magnet.org/GetAsset.action?contentId=783096&assetId=951521 (05/22/2007) .

[WP2 - 2006] WP2, *MAGNET Beyond Personal Network (PN) Architecture*, Copyright © 2006 Magnet Beyond, http://www.ist-magnet.org/-GetAsset.action?contentId=783096&assetId=951522 (05/22/2007) .

[WP6 - 2006] WP6, *Platforms for Personal Networks*, Copyright © 2006 Magnet Beyond, http://www.ist-magnet.org/GetAsset.action?contentId=783096&assetId=951530 (05/22/2007) .

D.3 Magnet related internal resources

[Schultz et al - 2006] N. Schultz, S. Tan, J. K. Sørensen, P. Karamolegkos, J. E. Larsen, L. B. Larsen, G. Nikolakopoulos, C. Olseni, C. Z. Patrikakis, C. F. Pedersen, J. S. Pedersen, M. Proschowsky, V. Protonotarios, R. Roswall, D. Saugstrup, L. Sørensen, S. Bessler, A. Hammershøj, E. Heinze, V. Kaldanis, D. M. Kyriazanos, J. Oksa, H. Olesen, and C. Xu, *Draft User Functionalities and Interfaces of PN Services (Low-fi Prototyping)*, Copyright © 2006 Magnet Beyond, Magnet Ref: IR1.4.1, Magnet Beyond.

D.4 Online resources

[MERL - 2008] *DiamondTouch*, Copyright © 2008 Mitsubishi Electric Research Laboratories, http://www.merl.com/projects/DiamondTouch/ (05/22/2007) .

[Magnet - 2006] *MAGNET Beyond*, Copyright © 2006 - 2008 MAGNET Beyond, http://www.ist-magnet.org (05/22/2007) .

[SUN - 2008] *Sun Java*, Copyright © 1995-2008 Sun Microsystems, Inc., http://java.sun.com (06/03/2007) .

[Walsh and Muellner - 2001] Norman Walsh and Leonard Muellner, *DocBook: The Definitive Guide*, Copyright © 1999, 2000, 2001 O'Reilly & Associates, Inc., 156592-580-7, O'Reilly, http://www.docbook.org/tdg/en/html (05/22/2007) .

[Wikipedia] *Wikipedia*, http://en.wikipedia.org/wiki (05/23/2007) .

Colophon

This report has been written using DocBook 4.5. It has been generated using Dblatex.