# USER HANDOVER IN A CROSS MEDIA DEVICE ENVIRONMENT

WILKO WIERINGA

A Coaching Service for Physical Activity

SUPERVISORS:
dr.ir. Rieks op den Akker
dr. Betsy van Dijk
Randy Klaassen, MSc
Harm op den Akker, MSc

LOCATION:
Enschede

# ABSTRACT

Multi-device systems can be used in different contexts, for example in a persuasive setting to help people change their lifestyle. It has been shown that by giving regular feedback, positive results can be reached for changing people's behaviour.

In a multi-device system the user should be able to interact with the device that is most appropriate for the interaction in a given situation. A way to ensure this is by using user handover between devices. The user object should be assigned to the device which fulfills a certain interaction the best. We can take the user handover a step further and also synchronize user information between the devices when a handover takes place. Now other devices also know the user information that resulted from dialogues they were not used for.

Inter-usability is about how easily users can reuse their knowledge and skills for an action when using a different device. Better inter-usability in a multi-device system enables users to switch more easily between these devices. It requires knowledge continuity and task continuity. Knowledge continuity is about how to retrieve and adapt knowledge or data which is managed with the system. Task continuity is about remembering the last operations done with the system, irrespective of which device this last operation is done with.

The goal of this project is to implement the user handover for a multi-device system and ensure inter-usability between the different devices in this system. This is done by implementing a multi-device system for helping office users in achieving and maintaining a healthy lifestyle.

This system is evaluated with six test participants, which demonstrated the correct working of the system.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LISTINGS

# ACRONYMS

ACSM American College of Sports Medicine

ADK Application Development Kit

API Application Programming Interface

C3PO Continuous Care & Coaching Platform

GPS Global Positioning System

GUI Graphical User Interface

KP Knowledge processor

MDS Multi Device Server

OIP Open Innovation Platform

R2D2 Roessingh Research and Development Database

RDF Resource Description Framework

RTMM Real Time Medication Monitor

SIB Semantic information broker

SOA Service Oriented Architecture

SOAP Simple Object Access Protocol

SSAP Smart Space Access Protocol

SRGS Speech Recognition Grammar Specification

TSS Text to Speech Synthesis

uPNP Universal Plug and Play

USB Universal Serial Bus

VoiceXML Voice eXtensible Markup Language

WHO World Health Organization

WSDL Web Service Description Language

# 1 INTRODUCTION

## 1.1 BACKGROUND

### 1.1.1 *Activity Promotion Systems*

The position of the American College of Sports Medicine (ACSM) on exercise for apparently healthy adults is to have at least 30 minutes of moderate exercise a day for at least five days a week [11]. The World Health Organization (WHO) state that thirty-one percent of adults globally were insufficiently active in the year 2008 [40]. In their Health Report of 2002 the WHO state that 1.9 million deaths can be attributed to physical inactivity and that it also costs 19 million healthy life years a day [39].

Kroeze et al. state that there is potential for using computer applications in education of physical activity and dietary behaviors [17]. Although Neville et al. report that the evidence for effectiveness of computer-tailored physical activity intervention is inconclusive [23]. Of the 16 articles they reviewed which described the evaluation of computer-tailored physical activity interventions, 10 of them found significant positive effects. The authors conclude that although computer-tailored interventions have the potential to reach large groups it is uncertain whether the reported effects are generalizable and sustained.

Hurling et al. measured the impact of a physical activity program, get*active! - described in Section 1.1.1 - which used a mobile phone and a web portal [13]. The test group had 77 healthy adults and the control group contained 30 healthy adults, which received no support. The average increase of physical activity of the test group over the control group was 2 hours and 18 minutes per week of moderate physical activity.

In the following part of this section, three Activity Promotion Systems will be described. Namely the system of the Smarcos project, the Continuous Care & Coaching Platform (C3PO) and the get*active! application.

#### Smarcos

The Smarcos project[1] tries to improve interusability of interconnected devices which communicate with their users. The project consortium consists of, among others, VTT[2], Philips Research Labs[3], Philips Con-

1 http://www.smarcos-project.eu/
2 http://www.vtt.fi/
3 http://www.research.philips.com/

sumer Lifestyle[4] and the University of Twente[5]. The Smarcos project receives funding from the European Union under the Artemis program[3].

The project tries to simplify the combining of multiple devices and/or services to create a new service, because in the current situation it is hard to make the interconnection of multiple platforms, services and user interfaces useful and enjoyable for users. To solve this, context-aware technology and model-based design is used [4].

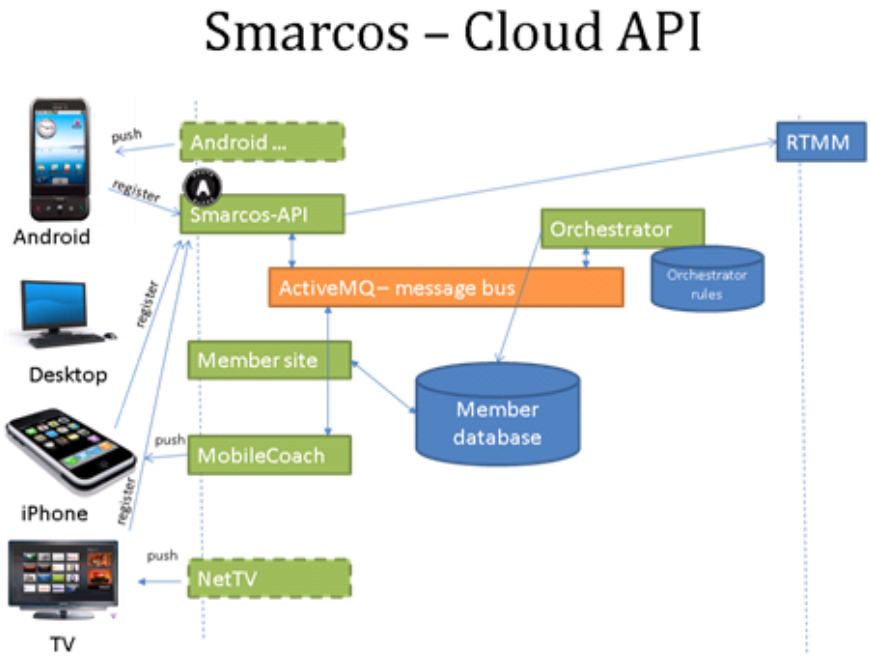


Figure 1: Smarcos cloud API

In Figure 1 the layout of the Smarcos cloud API is given. This figure shows how different devices work together, store information, retrieve information and how system wide events are generated. It works as follows:

- **ActiveMQ - message bus** This is an open source messaging and Integration Patterns server, which is developed by the Apache Software Foundation [1]. It provides the ability of communication between applications via messaging in an asynchronous, loosely coupled manner - which means that there are no interdependence or timing requirements.

  This kind of system is also called a message broker [33]. In the Smarcos system it takes care of routing information to devices or software components. Devices and software components can register with the ActiveMQ - message bus and indicate which kind of information or events they want to receive.

4 http://www.philips.com/about/company/businesses/consumerlifestylehighlights/index.page
5 http://www.utwente.nl/

- **Orchestrator** This software component is registered to receive all the information sent. It will write all the information to the Member database. Using a set of (predefined) rules, the Orchestrator can check if it needs to act and send for example a reminder. One example is when the user misses a medication event; the orchestrator will then send a medication reminder.
- **Member database** The database with all the information of the user/patient.
- **Member site** A web portal to see all the information of the user/patient.
- **Real Time Medication Monitor (RTMM)** A smart pill dispenser made by Evalan[6]. This system measures medicine intake and can send SMS notifications when a medication event is missed [26].
- **MobileCoach/Android/NetTV** This is the backend for the different devices and each device OS has its own backend. In order to communicate with the Smarcos-API each device communicates with its corresponding backend.
- **Smarcos-API** A device (backend) can register itself via the Smarcos-API. This works via Oauth authentication[7] which is an open protocol for secure API authorization [2].

  When a device has registered, the end user or the patient can give authorization to let the device access private data, in this case the member database.

Devices receive information via push messages and location or GPS information is sent back to the server. Work now is done to let devices send more information back to the server such as the level of glucose. This will be necessary to have more interaction with the user.

#### Continuous Care & Coaching Platform (C3PO)

The Continuous Care & Coaching Platform (C3PO) is a platform developed by Roessingh Research and Development[8] in order to remotely monitor and treat elderly patients and patients with chronic disorders. It is mainly focused on continuous monitoring and providing feedback to the users [25].

In Figure 2 [25] the architecture overview of C3PO can be seen. The framework has four different components; sensor, smartphone, server and web portal. This setup allows the usage of multiple and different sensors. The four different components work as follows:

6 http://www.evalan.com/
7 http://oauth.net/
8 http://www.rrd.nl/

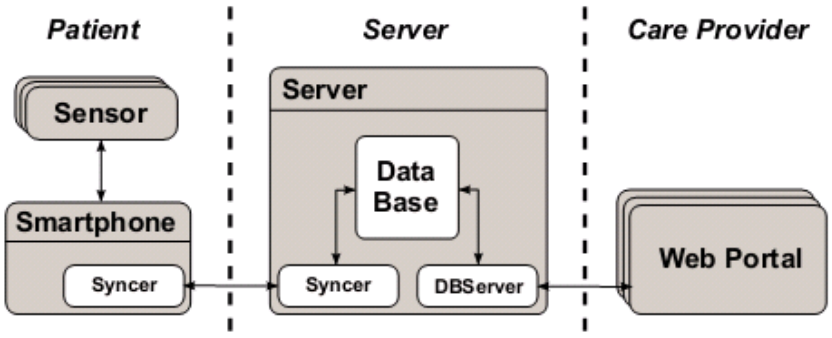


Figure 2: High level architecture overview of C3PO

- **Sensors** A sensor node captures, processes and sends information. Currenly the ProMove-3D wireless sensor is used, which has an accelerometer, gyroscope and magnetic compass.

- **Smartphone** A software framework runs on the smartphone. Applications can be made by combining several modules, where each module performs a small task.

  The modules communicate via a central module called the Hub, to which they can register and receive notifications for new information.

  In the current setup there is one module which connects to the sensor node via Bluetooth, it outputs the data of the sensor node and keeps track of the status of the connection. Another module, AndroidSync, is responsible to synchronize the data between the smart phone and the server.

- **Server** The core for the server is the Roessingh Research and Development Database (R2D2), which supports a variety of data types. Supported data types are for example bio signal data, questionnaire results, context data and subject information.

  The data is stored structurally which makes it possible to compare the data of different studies. The data is structured in such a way that all measured data links to a subject and a timeline. The various activity data of a subject are linked via the time the activity was done.

  A toolkit, the RRDToolkit, is developed in order to visualize all the data of a subject. The user's access of data sets requires authorization.

- **Web portal** A modular framework is created to create web portals, which shares a similar modular setup compared to the Smartphone framework. Via the web portal biomedical researchers, patients and health care professionals can view the data in an understandable way.


Figure 3: Weekly schedule for planning physical activity

**get*active!**

get*active! is a system developed by Unilever[9] and Tessella[10], although it is only used in test trials. In the annual report of 2006-2007 it was reported that the system was moved from Unilever Corporate Research to a newly created business [34], although nothing can be found about this company or about the system being in use.

In the get*active! application the activity of the users is measured with a wrist-worn accelerometer. At the start of the program users can identify perceived barriers to physical activity with the online application. Once a week the participants need to complete a questionnaire regarding their exercise level during the week. After which they were given constructive feedback based on their target and performance relative to the other participants. The application contains a library of different physical activity options [13].

In Figure 3 [13] the schedule can be seen where the physical activity sessions can be planned. The schedule is also used to send email or SMS reminders of the activity when the user indicated he would like this [13].

After filling in the schedule an automated assessment tool provided feedback on the amount and type of activity planned. It warned the users when the amount of activity was much larger compared to the weeks before and then it would recommend a reduction of physical activity [13].

Charts are displayed for the current day, the current week and overall. In these charts it is visible how active the users were. The bands

9 http://www.unilever.com/
10 http://www.tessella.com/

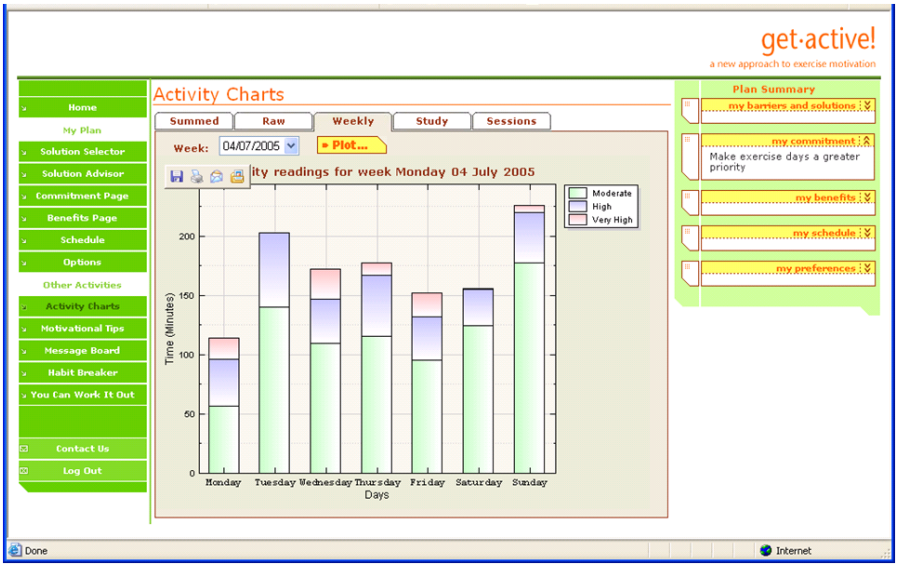


Figure 4: Weekly activity charts

of the charts are divided in moderate, high or very high performance. An example of this screen can be seen in Figure 4 [13]. The application provides motivational tips based on the current activity level and when the user missed a planned activity the application tries to find the reason behind this.

### 1.1.2 *Cross Media Systems*

Cross media systems are, according to Segerstahl [31], composed of a set of distinctive devices and applications. The main goal for these devices and applications is to optimize communication and interaction in different contexts for a specific human activity [31].

Segerstahl states the following main challenges for cross media systems [31]:

- **Heterogeneity** Users need to have higher technological skills when dealing with several devices and applications. Different kinds of media may also evoke different expectations by users.
- **Interoperability** Devices are interconnected with each other, but this is not the only interoperability in a cross media system. The conceptual of functional architecture of the system plays also an important role. The architecture determines how well the system performs with regards to the interoperability.
- **Consistency** When a cross media system is developed by multiple companies or organizations, there is a chance of inconsistency regarding terms used and interaction logic.

Boumans ([7], as cited by Segerstahl [31]) gives five key characteristics of a cross media system:

1. More than one medium is involved in supporting a message/story/goal
2. The aim is on integrated production of support functionality
3. Content is delivered on multiple devices
4. More than one medium is needed to support the *whole* message/story/goal
5. The common message/story/goal is spread on the different platforms

In the following part of this section, a Cross Media Middleware solution and a Cross Media System will be described. Namely the Smart M3 - Sofia framework and The Polar Fitness System.

### Smart M3 - Sofia framework

The Smart M3 - Sofia framework is not built around a specific human activity, but is a middleware solution for realizing cross media systems. Liuha et al. [19] describe that the SOFIA project exists for creating smart environments with the Smart M3 - Sofia platform. In order to create a smart environment the functionality presented to users needs to be uniform and predictable. Furthermore, the operation of the different devices needs to be the same as well.

The Sofia project works with a smart space, which is an ecosystem of interacting objects. Physically these are formed by sensors, devices and appliances which are self- organizable and give services and data to a person. This is in line with the pervasive computing vision. A smart space needs to deal with dynamicity, scalability, trust and privacy [19].

Kiljander et al. [15] state there is not a standard definition for smart spaces, but there are some common features and attributes:

- It is a physical space enhanced by different types of connected devices.
- Smart space has means of collecting information of entities in the space, for example sensors, cameras and user interfaces.
- Devices should be able to perform actions that are relevant for the given scenario without user interaction.

Kiljander et al. [15] give the following definition:

> The Smart Space is an information environment and a set of interoperable devices in some physical space that enables the selected exploitation of smart services to the user.

The most known application domain independent solutions for interoperability are:

- **Web services** Web services implements a service-oriented architecture (SOA) and provide a standard way for applications to interact with each other. It is based on two standards, Web Service Description Language (WSDL) and Simple Object Access Protocol (SOAP).
- **Universal Plug and Play (uPNP)** uPNP is, according to Kiljander et al. [15] "a group of networking protocols that aims to provide a distributed architecture for pervasive peer-to-peer network connectivity for intelligent devices."
- **The Semantic Web** The Semantic Web, according to Kiljander et al. [15], "approaches the information level interoperability by presenting the information of the Web using ontologies."

Kiljander et al. [15] say that with Smart Spaces there is no "aim to provide interoperability by defining service interfaces for specific use cases", this in contrast with Web services and uPNP.

Liuha et al. [19] state that for the Sofia project "[t]he main mission is to make in-formation in the physical world available for smart services in embedded and ubiquitous systems." Practically, the Sofia project develops an Open Innovation Platform (OIP) architecture (Smart-M3) and Application Development Kit (ADK).

#### Smart-M3 - Sofia

Liuha et al. [19] say that device interoperability can be divided in three layers:

- **Device world** The transfer of bits between devices.
- **Service world** The interoperability between applications on different devices.
- **Smart world (or information level)** The meaning of information is the same on the different devices.

Smart-M3 focuses only on the information level, the level which makes the information available. The other two levels are solved by using existing technologies. The information level is solved in the following way [19]:

- **Semantic information broker (SIB)** The SIB is used to store, share and govern information of one smart space. A smart space can exist of multiple SIBs and the information is stored in a Resource Description Framework (RDF).
- **Knowledge processor (KP)** The KP processes information and stores and or consumes information from a SIB.

- **Smart space access protocol (SSAP)** Protocol for communication between KP and SIB.

Liuha et al. [19] state that Smart-M3 is not meant to send commands between devices or KPs, this should be done using service level capabilities. The KPs and SIBs need to have a common ontology model, data format and information access solutions in order to have smart space solutions. A smart space can exist of multiple SIBs in order to allow separation of information, for example a SIB for personal and family information and a SIB for weather services.

According to Honkola et al. [12] information consistency, regarding the structure or the semantics of the information, is not guaranteed, with which Honkola et al. [12] mean that the agents can interpret the information in the way they want. The reasons for this are responsiveness of the SIBs, because it lessens the computational burden of the infrastructure. The other reason is to let KPs freely mix the data and thus not stiffen creativity. Otherwise new ways of combining data would require approval of an ontology governance committee.

Honkola et al. [12] are planning to create a mechanism to attach agents directly to a SIB in order to add more complex reasoning, for example to repair the ontology or to translate ontologies. These agents would have an interface to the SIB which allows them to do more than normal KPs, but these agents are not permitted to join another SIB or smart space.

Kiljander et al. [15] state that there are two fundamental models for Smart-M3-based Smart Spaces, centralized smart space and distributed smart space. In the centralized model there is a single SIB which conveys the information in the smart space to the KPs. With the distributed model, there exist many devices with a SIB. This setup can be used if a device cannot be dependent on an external SIB and transaction between KPs is faster if they are on the same device. For embedded systems Kiljander et al. [15] use the central model, because:

- A SIB needs to store large graphs and perform heavy computing.
- The current implementation of a SIB cannot be easily ported to an embedded device.
- The distributed model requires synchronization of information which could lead to poor performance in embedded systems.

### The Polar Fitness System

The polar fitness system[11] is a system which is intended for ordinary fitness-minded people [32]. It consists of a wearable heart rate moni-

11 http://www.polarpersonaltrainer.com

tor, GPS receiver, a USB dock for transferring data and a web portal, see Figure 5 [31].


Figure 5: The polar fitness system: wrist unit, website and data collection accessories

A fitness program with weekly targets is provided by the heart rate monitor. It gives the user information and instruction during exercises and help the user into getting the correct heart rate. Feedback is provided after each exercise and a weekly summary is given which also contains suggestions for the following week. The website contains:

- a training planner
- tools for managing long term training and detailed exercise programs
- information about and instructions for heart rate based exercise
- progress charts and graphs
- summaries of analysis and follow ups
- document the users exercises
- storage for exercise data

The user plans his exercise with the website, performs the exercise and receives help from the monitor and finally he can have a follow up on the website.

Segerstahl found out after testing that the user's primary motivation to exercise as well as their willingness to use different devices determined how much of the system's components were used [32].

In the same evaluation it came to light that the structural and compositional architecture of the system influenced which training methods are supported.

In a previous study, Segerstahl reported that many users abandoned the website [31]. The usage behavior of users corresponded to the mental representations the users had of the system. The adaptation as well as the mental representations were influenced by how well the different media fitted the users primary activity and goals, and how well the system supported coherent user experiences and interusability. She concluded that the system did not promote the interusability between the monitor and website enough [31].

## 1.2 GOALS & CHALLENGES

One of the main goals for this thesis work will be to extend a single device activity promotion system, the Continuous Care & Coaching Platform (C3PO), to a cross media or multi device system. For more information about C3PO, see Section 1.1.1.

When working with a multi device system we do not want to confuse the user by interacting with him on multiple devices at the same time. Therefore only one device at a time can interact with the user and in order to accomplish this, we come to another main goal. This is user handover, which means when a device wants to interact with the user he should have the user object. A device can request the user object and when this request is granted by a main server, a handover of the user object takes place.

The third goal is to have dialog abilities in the system, we want to let the users fill in questionnaires. This of course in the context of a cross media system with user handover, which should be incorporated in the dialog system and work naturally for the user.

The challenges we will encounter will be partly related to a cross media system. In Section 1.1.2 some of these challenges were already mentioned, .

Segerstahl main challenges were already mentioned in Section 1.1.2, but we will repeat them here shortly:

- **Heterogeneity** Higher technological skills required of users when dealing with more devices.
- **Interoperability** The interconnection between devices and interconnection representation in the architecture itself.
- **Consistency** Consistency in terms used and interaction logic.

Oquist et al. [27] also give the following usability challenges which are of importance for a cross media system:

- **Portability** How portable is the device which should be carried around. In our case it is the smartphone, although many differ-

ent smartphones can be used in our system we can still use one specific smart phone and see how easily people carry it around when they are leaving for example their workplace.

- **Attentiveness** Oquist et al. state that when portability of a device increases, the amount of attention it can demand goes down. An example in our case would be if users would notice that they receive a questionnaire when they are walking. How easy is it for them to fill in that questionnaire. Does it help users that they can switch the modality or realizers when they receive a questionnaire?

- **Manageability** How easily can users manage the GUI of the different devices. In the case with the smartphone, can the user use only one hand to interact with the application.

- **Learnability** How easily can the user understand and use the different functionalities and, when switching between devices, does the user experience it as the same interaction paradigm.

More usability challenges for cross media systems, or multi device systems, are given by Denis et al. [10]. They coin the term inter-usability and with this they mean how easily users can reuse their knowledge and skills for an action when using another device (see also Section 2.4). Inter-usability has two dimensions [10]:

- **Knowledge Continuity** How easily can users understand how to perform certain functions supported by the system with the different devices. Ideally this is the same for each device, although Denis et al. acknowledge this is not feasible, given the constraints of mobile devices. Knowledge continuity usability points to look at are:
  - **Visual appearance and continuity** Can the user find the same service in the same place and does he think the interface looks the same. For example, do buttons on different devices with similar functionality have the same label?
  - **Partition of Data and Functions** If there is a discontinuity between devices, a particular service is not available for its entirety on some devices, is this apparent to the user?
  - **Procedures** Same functionality on different devices should be accomplished in the same way.

- **Task Continuity** All the different devices are aware of the actions done in the system by the user, irrespective with which device the action was performed. Task continuity usability points to watch are:

- **Recovery of State of Data** When an activity is interrupted and is resumed on another device, the state of the data should be presented in such a way which confirms to the expectations of the user.
- **Recovery of Activity Context** When a user interrupts his activity, he leaves at a specific point in the application - for example the cursor in a text editing program at the end of an incomplete sentence. When he resumes the activity after a long enough period, he can forget his last action when the activity context is not saved. For example, the cursor is after the resume of the activity at the beginning of the document and the user forgets he was finishing a sentence. This is also a problem for one-device applications, but a multi device system can aggravate the problem.

## 1.3 OUTLINE OF THE THESIS

In the next chapter, Chapter 2, we lay down the requirements for the system. This is done via a scenario and existing literature. In this chapter the assumptions we employ when developing the system are also described.

In Chapter 3 we give some background on dialog systems and what our dialog architecture is.

After the Dialog chapter, we will continue with Chapter 4 System Architecture. In this chapter we describe how we extended the Continuous Care & Coaching Platform (C3PO), what kind of devices we have, the GUI's we use and the network setup. We finish with an overview of the complete architecture.

Chapter 5 is called System Implementation and it will follow the overview given at the end of Chapter 4, but it will explain how things are done.

The next chapter is Chapter 6 Evaluation. Here we describe how we evaluated our prototype and what the results of the evaluation were.

We finish this thesis with Chapter 7 Conclusions. Here we discuss how we solved or taken on the goals & challenges, stated in Section 1.2. The chapter is finished with a section on Future Works, what are the possible next steps in lifting the C3PO cross media system to an even higher plan.

# 2 REQUIREMENTS

In this chapter we will come to a list of requirements to which the system must adhere or which it must deliver.

We need to make a distinction between requirements for the whole of the system and requirements for a specific application. With this we mean that we will look how it is possible to create a space for a persuasive system. We will not look, for example, how a persuasive dialog should take place, but how we can enable that it can take place in the first place.

First we will give a scenario in how the system can function, after which the use cases of the system will be given. With the help of the scenario, the use cases and with the help of design strategies for persuasive technologies proposed by Consolvo et al. [8] the requirements for the system will be determined. After each requirements list, we will give the reason for those requirements.

When the system is mentioned in the requirements, the whole of devices, the server and the applications running on the devices is meant.

## 2.1 SCENARIO

Mr. John Johnson is 35 years old, a little bit overweight and works at his current company, Initech, for ten years. Initech takes the interests of its employees to heart and takes the American College of Sports Medicine (ACSM) position[1] seriously. They have decided to purchase a system that helps their office workers in achieving and maintaining a healthy lifestyle, inside the office, but also in their private lives. The system consists of a smartphone - which tracks the amount and time of activities done, the users' desktop computer and monitor and a computer and monitor at the coffee corner. The employees are given feedback via their desktop computer screen, mobile phone and a screen at the coffee corner of the company.

John arrives at work and works for two hours after which he stares a bit out of his window. A message with a graph pops up on his desktop screen. The graph shows John's activity so far, with a projection on his daily goal, the message is an advice and contains the following: "I see that you have meetings in the afternoon; now would be a good time to go for a lunch walk". This seems like a good idea to John and he goes for a walk after half an hour.

1 See Section 1.1.1

After his lunch, John gets a cup of coffee at the coffee corner. He chats a bit with his colleagues and looks at the screen hanging besides the coffee machine. It displays the name of Paul, which is the person the most active that day, compared to the people in the vicinity of the coffee machine. John is a bit disappointed it isn't his name.

A full day of meetings and work has passed and John drives back home in his car. When he is at home, he cooks a meal and after he finished it he sits on the couch watching television. Suddenly his phone vibrates and after taken the phone out of his pocket he sees a message urging him to do a 10 minute walk around the block in order to reach his daily quota. John thinks he can squeeze that in before the start of his favorite television program and takes the walk. After the walk and the television program John goes to bed, knowing he has done enough exercise for that day.

The next day John looks at his activity history online via his web browser. He is proud of himself, because he sees the progress in the days he reached his daily quote. John wonders if Paul is also doing so well and tries to find his data, but the website only allows him to see his own data.

## 2.2 USE CASES

In the following section use cases will be given of the system. As Maciaszek states, use cases represent a complete unit of functionality of value to an actor [20]. Use cases can communicate with an actor, but this is not necessary.

Our use case description will contain the following [20, 28]:

- **Brief description**
- **Actors** involved
- **Preconditions** necessary for the use case to start
- **Detailed description** of flow of events that include:
  - **Main flow** of events, that can be broken down to show:
    - **Subflows** of events
  - **Alternative flows** to define exceptional situations
- **Postconditions** that define the state of the system after the use case ends

| **Use case 1** | **Device sign on** |
|---|---|
| Brief description | This use case allows a device to sign on, on the server |
| Actors | User |
| Preconditions | Device is not registered at the server.<br>Device has an Internet connection. |
| Main flow | The device is turned on by the user and the local application is started. The application registers itself with the server and receives an acknowledgment.<br>When the smartphone device signs on and this is the first device to sign on, it receives the user object. See also Table 3. |
| Alternative flow | The server is down and the device is the smartphone.<br>When this happens the application will start and by definition the smartphone has the user object.<br>When the server comes online the smartphone registers itself and start synchronizing.<br>The server is down and the device is the kiosk or desktop.<br>When this happens the application will not start.<br>When the server comes online, the device will register itself and start synchronizing. |
| Postconditions | When the use case was successful, the device is registered at the server and can receive the user object.<br>When not successful only the smartphone can have the user object if it signed on. |

Table 1: Use case 1: Device sign on

| **Use case 2** | **Device sign off** |
|---|---|
| Brief description | This use case allows a device to sign off from the server |
| Actors | User |
| Preconditions | Device is registered at the server. |
| Main flow | The device is turned off by the user and the local application is shut down.<br>The application unregisters itself with the server and receives an acknowledgment.<br>When a device signs off which has the user object, the user object is handed over to another device if available.<br>If the smartphone is still registered, the smartphone will receive the user object. |
| Alternative flow | None |
| Postconditions | When the use case was successful, the device is unregistered at the server.<br>If it has the user object and there are other devices still registered, the user object is transferred to another device.<br>The smartphone has the highest priority in receiving the user object after a device signed off. to sign on. |

Table 2: Use case 2: Device sign off

| **Use case 3** | **User handover** |
|---|---|
| Brief description | This use case allows a user handover between devices. |
| Actors | User |
| Preconditions | At least two devices are registered at the server and one device has the user object. |
| Main flow | In general terms the main flow goes as follows.<br>Device A has the user object.<br>The user moves or does another action and device B determines the user is in its reach and request the user object from the server.<br>The server approves and assigns the user object from device A to B.<br>An example is the following:<br>The smartphone has the user object and the user walks with his smartphone to his desktop computer.<br>When he starts using the computer, the local application on the computer requests the user object from the server.<br>The server approves and the desktop computer has the user object. |
| Alternative flow | Device A has the user object, Device B requests the user object from the server.<br>The request is denied. |
| Postconditions | When the use case was successful, the device which had the user object does not have the user object anymore.<br>The requesting device now has the user object.<br>When the use case was unsuccessful, nothing is changed. |

Table 3: Use case 3: User handover

| **Use case 4** | **Feedback** |
|---|---|
| Brief description | This use case allows a user to receive feedback on his performance. |
| Actors | User |
| Preconditions | At least one device is registered at the server and one device has the user object. |
| Main flow | The user can receive feedback on the device which has the user object, in the following circumstances:<br>• the user under performs (too much or too little activity)<br>• the user reaches his daily goal<br>• summary feedback, determined by the running schedule.<br>The user than can read the feedback. |
| Alternative flow | None. |
| Postconditions | A feedback message won't be given anymore for a set time for under performing if the message was triggered by under performing.<br>The scheduled feedback message won't be given again when it was a scheduled feedback.<br>A daily goal message won't be given anymore if it was a daily goal message. |

Table 4: Use case 4: Feedback

| **Use case 5** | **Questionnaire** |
|---|---|
| Brief description | This use case allows a user to receive a questionnaire and fill this in. |
| Actors | User |
| Preconditions | At least one device is registered at the server and one device has the user object. |
| Main flow | The user can receive the questionnaire on the device which has the user object, when this is triggered by a set schedule.<br>When the questionnaire starts, the user receives the first question.<br>After answering the question, the following things are possible:<br>• Next question, based on question list.<br>• Next question, based on given answer.<br>• End, based on question list.<br>• End, based on given answer.<br>The answer is processed by the device and also synchronized with other devices registered at the server.<br>The form in which way the question is presented and in which way the answer can be given is determined per device. |
| Alternative flow | None. |
| Postconditions | The questionnaire triggered for the specific schedule, will not be triggered again.<br>Other devices are aware of the answers given by the user. |

Table 5: Use case 5: Questionnaire

| **Use case 6** | **Realizer switch in questionnaire** |
|---|---|
| Brief description | This use case allows a user to switch in and output realizers in a questionnaire. |
| Actors | User |
| Preconditions | A device takes a questionnaire with the user. |
| Main flow | When the user is presented a question of the questionnaire, it has the possibility to switch between the available input and output realizers. When an output realizer switch is done, the current question is repeated with the new output realizer. When an input realizer switch is done, the answer can be given with the new input realizer. |
| Alternative flow | When the switch was not possible, the current realizer is retained. |
| Postconditions | The next questions and questionnaires are done with the selected input and output realizers. |

Table 6: Use case 6: Realizer switch in questionnaire

| **Use case 7** | **User handover during questionnaire** |
|---|---|
| Brief description | This use case allows a user handover between devices during a questionnaire.<br>In effect this makes it possible to continue the questionnaire on another device. |
| Actors | User |
| Preconditions | A device takes a questionnaire with the user and at least one other device is registered with the server. |
| Main flow | During a questionnaire it is possible that another device requests the user object.<br>When this request is granted by the server, the other device will receive the user object.<br>When it notice a questionnaire is active, it will continue the questionnaire at the point the switch happened.<br>Also it is aware of previous answers. |
| Alternative flow | When the user handover was not possible, the current device retains the user object and nothing changes.<br>The user is not aware of anything. |
| Postconditions | The new device has the user object until a new user handover takes place. |

Table 7: Use case 7: User handover during questionnaire

| **Use case 8** | **Questionnaire reminder on kiosk** |
|---|---|
| Brief description | This use case allows a reminder to be displayed on a kiosk device, when the user has to finish a questionnaire and when he is in range of the kiosk device. |
| Actors | User |
| Preconditions | A device takes a questionnaire with the user and at least one kiosk device is registered with the server. |
| Main flow | It is possible for a user not to answer a questionnaire, in effect postponing it.<br>When this happens and the user walks by or is in reach of a kiosk device, the kiosk device will display a reminder. |
| Alternative flow | None. |
| Postconditions | Nothing has changed. |

Table 8: Use case 8: Questionnaire reminder on kiosk

## 2.3 FUNCTIONAL REQUIREMENTS

According to Maciaszek functional requirements, or service statements as Maciaszek names them, are about the expected services of a system. What a system must accomplish. The functional requirements can be subdivided in requirements that describe the scope of the system, the necessary business functions and the required data structures [20].

A note regarding the requirements is the following; with the system, the whole of devices, the server and the applications running on the devices is meant.

1. The system should know where the user is.
2. The system should know where the devices are or which devices are in the vicinity of the user.
3. The system should be able to read a questionnaire specification and perform that questionnaire.
4. The system should be able to give feedback
5. The system should make it possible to switch between different kind of input and output modalities.

Functional requirement 1 (F1) and 2 (F2) stem from the fact that the system is about interacting with a mobile user via multiple devices (non-functional requirement 2). Without F1 and F2 the system cannot

decide which device is currently the best option to interact with the user.

F3 and F4 make it possible to have an activity promotion system, where users can receive feedback and state through questionnaires how they are doing.

F5 makes it possible for the user to receive and give information in the way it suits him the best. Making it possible

## 2.4 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are, according to Maciaszek, statements about how the system is constrained in accomplishing its services [20]. Maciaszek calls non-functional requirements, constraint statements. The non-functional requirements can contain statements about the look and feel of the system, performance, security, legal, etcetera.

1. The system should be able to always interact with the user.
2. The system should support the usage of multiple and different devices.
3. Only one device at a time can interact with the user.
4. The system should make it possible to answer a questionnaire on multiple devices, but not at the same time.
5. The system should enable that all devices have access to the latest data of the user when necessary, in other words data should be synchronized between devices.
6. It should be possible to schedule questionnaire and activity feedback
7. It should be possible to perform the functional requirements with only the smart phone

Non-functional (NF) 1 supports the Goal-Setting Theory which, among other things, states that not only when the goal is met feedback should be given, but also when progress is made[8]. NF 1 also supports the Cognitive Distance Theory, which states that when attitudes and behavior are inconsistent the individual wants to eliminate or reduce this dissonance. Consolvo et al. state that in order to help the individual with their commitment to change

> "[t]he awareness provided by the technology should be persistently available and easy to access [8]."

It is possible to give the user always feedback on his smartphone, but this would limit the form in how we can give feedback or interact with the user. With NF2 it is also possible to create other ways of

doing this. This also enables the Aesthetic design goal of Consolvo et al., which states the display should be inquisitive and sustain interest. By using different display devices the user is less easily bored and thus will sustain interest.

NF 3 makes it possible to not confuse the user, which can happen when multiple devices would want to interact with the user. Especially for older people, which are the main target of an activity promotion system.

NF 4 makes the Trending / Historical design object possible of Consolvo et al, which states that the user should be given information about his past behavior as it relates to his goals [8].

The other design objects of Consolvo et al [8], Abstract, Non-Intrusive, Public, Positive and Comprehensive are not the responsibility of the system - they should be done by individual applications. Every design object is also not made impossible by these requirements for the system architecture.

Denis et al. state that inter-usability of a multi-device system is important for users, because it enables them to transition more easily between devices [10]. With inter-usability is meant how easily users can reuse their knowledge and skills for an action when using another device. They introduce two kinds of continuity which a multi-device system should take into account; knowledge continuity and task continuity.

Knowledge continuity is about how easily users can understand how to perform certain functions supported by the system with the different devices. According to Denis et al. ideally the representation or functionality of a specific service of the system should be the same for all of the devices [10], although this goal is not realistic given the constraints of mobile devices.

Task continuity is about remembering the last operations done with the system by the different devices, irrespective of which device this last operation is done with.

Knowledge continuity and task continuity is made possible by NF 4 and 5, although the specific applications should take the two kinds of continuity into account as well.

NF 6 makes it possible to create an activity promotion application, were users can be given feedback on the progress and their state can be measured.

NF 7 enables NF 1, NF 7 is also specified by Roessingh Research and Development, creators of C3PO, because this is part of their application and design philosophy.

## 2.5 USER REQUIREMENTS

This section states the requirements the user should fulfill or, in other words, who is our target user.

1. The user should have a basic understanding of working with smartphones and computers.
2. The user should take the smartphone everywhere he goes.

Both user requirements are the minimum requirements to make the system successful. Without UR 1 the user could do nothing with the system or understand how to react to questions or feedback.

UR 2 is necessary in order to always fall back to the smart phone when there is no connection. It also makes it possible for devices to detect the user's presence. The third result is that the user can always receive feedback or a questionnaire when possible, fulfilling non-functional requirement 1.

## 2.6 ASSUMPTIONS

In this section the assumptions with which the system will be developed will be stated.

### 2.6.1 *Internet*

We will assume that each device has a working Internet connection. The reason for this assumption is to focus on the more interesting parts for a multi device system. When this system should be used for real users, this cannot be assumed of course. After the prototype this should be taken into account and the connection modules should have added functionality.

### 2.6.2 *Security*

There will not be any security in the prototype. The reason is to focus on more interesting parts for a multi device system. With security is meant the connection of a device to the server and that it is not possible to see sensitive information of users.

### 2.6.3 *One user*

The prototype should make it possible to support multiple users later on. When we start with the system we will assume there is only one user.

### 2.6.4 *Clocks*

The clocks or time on the different devices are the same.

### 2.6.5 *Smartphone*

We assume the user always has the smartphone with him.

## 2.7 CONCLUSION

We now have a list of requirements which the system should deliver or adhere to. In Chapter 3 and Chapter 4 it will be explained how the system will fulfill all the different requirements determined in this chapter.

# 3 DIALOG

This chapter describes the background of dialog and dialog systems. The dialog system in our system will make it possible to ask the user questions for the questionnaire. The dialog system will fulfill Functional Requirement 3 (The system should be able to read a questionnaire specification and perform that questionnaire), for more information about the requirements see Chapter 2.

First an overview of the current state of the technology is given. This is followed by the architecture we will be using for our system and the different input and output realizers are described. We will finish with a section about health dialog systems and health dialog, is there something special about health dialog compared to regular dialog?

## 3.1 DIALOG SYSTEM TECHNOLOGIES

In this section an overview of available dialog system technologies is given.

### 3.1.1 *Pattern-response Dialog Systems*

A pattern-response dialog system gives an output based on a rule. A sequence of words given by the user is matched against the rules which the system has. When a match is found, the rule is evaluated and the resulting output is given. There is no state of the dialog retained and the system also does not understand what is said, it merely reacts [37].

An example of such is system is the famous ELIZA program, developed by Joseph Weizenbaum and is one of the first natural language processing programs, albeit in a primitive form. It uses simple pattern matching to give a response to a certain output [37].

Because such a system does not understand what is said, but merely reacts, this is the reason why a pattern response system cannot be used in a critical application where errors can have a serious impact [6].

### 3.1.2 *State-based Dialog Systems*

A state-based system represents the dialog as a finite state machine , where each dialog move is represented by a state and each possible state transition is represented by an arc [14].

When each state has as a maximum one resulting state the system is called state-based linear. When for example an answer determines the next state and there are more possible states than one the system is called a state transition network.

An extended version of the state transition network is the hierarchical state transition network, where a certain state can be a sub-dialog. This sub-dialog is the same every where, but it can be invoked at different places (or states). This can be compared to sub-routines or function calls in a programming language.

When the dialog system also uses information retrieved from a database or it stores information from the dialog in a database and it uses the information from the database to make a decision which transition to chose, the dialog system is called an augmented transition network.

Augmented transition networks where originally developed for sentence parsing and are the most used systems for a health dialog system [38, 6].

### 3.1.3 *Plan-based Systems*

Plan-based systems try to infer the user's goal and task because the things people say do not always correspond to what they intend. These kind of systems assume that the speaker's speech acts are part of a plan and the system should respond to this plan instead of that what is said by the user.

This plan inference can be computational complex and are not yet used much in health dialog systems [6].

## 3.2 DIALOG INITIATIVE

In a dialog system there is support for one or more types of initiatives. Namely system initiative, user initiative and mixed initiative.

System initiative - also called single initiative - means that the dialog system is in complete control of the conversation [14].

With user initiative the user starts and controls the conversation at anytime he wishes.

In the case of mixed initiative, the control is shared. The system can ask questions when it needs information or clarification, as well as the user can [21].

For our system we will create a dialog system which is system initiative based, this because we only need to take questionnaires.

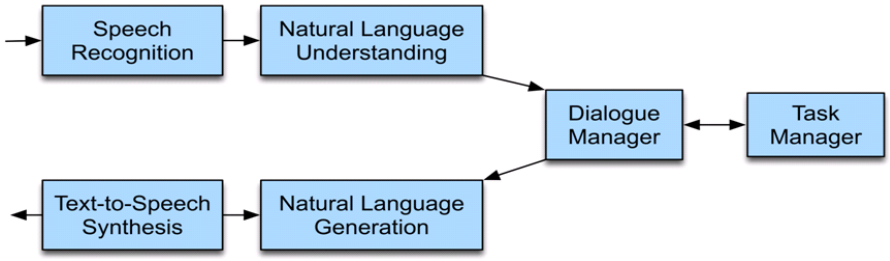


Figure 6: Standard Architecture for a Dialog System

## 3.3 DIALOG SYSTEM ARCHITECTURES

In Figure 6 [14] a standard architecture for a dialog system is given and a description for the components, according to Jurafsky et al., is as follows:

- **Speech Recognition** Analyzes the user's speech and converts it to a string. Instead of speech it could also be a handwriting or gesture recognizer.
- **Natural Language Understanding** This parses the string in such a way that the system knows what is said, it represents the meaning of the sentence. How this is done and the results of this are depended on the system. It can be done with name identification, part of speech tagging and / or a semantic parser.
- **Dialog Manager** This component controls the flow of the dialog. With the input given it determines how the conversation should proceed, when and what questions or statements to make. The Dialog Manager also maintains the state of the dialog.
- **Task Manager** The Task Manager interfaces with the Dialog Manager and has knowledge about a specific task.
- **Natural Language Generation** In this component the syntactic structures and words are chosen. In it simplest form all the words in a sentence are prespecified. Another simple form is template based generation where sentences can have variables which can be filled in. A more sophisticated and complex method is to build or generate a sentence based on the meaning or intent given by the Dialog Manager.
- **Text to Speech Synthesis (TSS)** This component outputs the speech based on the given sentence. This could also be an animated avatar.

Not all dialog systems have these components, it is possible that a dialog system has the Natural Language Understanding, Dialog Man-

ager and Natural Language Generation components. Such a dialog system can read text from the input and gives text as output.

## 3.4 EXAMPLE OF A DIALOG SYSTEM ARCHITECTURE

In this section we will give an example of a Dialog System Architecture.

### 3.4.1 *VoiceXML*

VoiceXML stands for Voice eXtensible Markup Language and is an XML based dialog design language [14]. With VoiceXML it is possible to create voice-user interfaces and just like going to a website with a browser it is possible to access a VoiceXML application with a telephone via a VoiceXML interpreter, which is called a browser, which resides on a telephony server [18]. The model underlying VoiceXML is an augmented transition network [6].

```
<field name = "to">
 <prompt>
  Welcome to Ajax Travel. Do you want to fly to
  <emphasis>New York</emphasis> or
  <emphasis> Washington </emphasis>
 </prompt>
 <grammar type="application/grammar+xml" version="1.0" root="city">
  <rule id = "road" scope = " public">
   <one-of>
    <item> New York </item>
    <item> Big Apple <tag> New York </tag> </item>
    <item> Washington </item>
    <item> The Capital<tag> Washington </tag></item>
   </one-of>
  </rule>
 </grammar>
</field>
```

Figure 7: Example of a VoiceXML field

In Figure 7 [18] an example of an VoiceXML field can be seen. The green part is the Speech Synthesis Markup Language, which specifies the speech, volume, inflection, and prosody of the the text. This is converted by a Text-To-Speech Synthesis engine to acoustic speech. Emphasis is given to the words "New York" and "Washington".

The red part in Figure 7 is the Speech Recognition Grammar Specification (SRGS) and specifies the words and phrases where the system listens to. There are two formats for SRGS, XML and Augmented Bakus Naur Form.

The blue part in Figure 7 is the Semantic Interpretation (SI) and the language is based on the ECMAScript. With the SI it is possible to derive the semantic meaning of text or speech. It is often used to extract words and translate them into an internal representation. In this case "Big Apple" should be interpreted as "New York".

## 3.5 INPUT MODULES FOR DIALOG

For our system the following input modules are created:

- **Text input** This module can receive and process text input.
- **Speech input** This module can receive and process speech input.

When the Dialog Manager sends the question, it can also send possible answers - making it a multiple choice question. When it does not send any answers it is an open question, in this case it sends the type of the answer it expects - for example a number type to specify it expects a number.

When we look at Figure 6 the Realizer is Speech Recognition, Natural Language Understanding, Natural Language Generation and Text to Speech Synthesis components, depending on the kind of in- and output it supports.

When the Realizer has outputted the question, received the input and determines the meaning of the input, it sends the answer back to the Dialog Manager. The Dialog Manager will then select the next question based on the state it is in. Is there a conditional arc, is there an unconditional arc or is the dialog at its end.

Text input will be supported by the desktop and smartphone device, speech input only by the smartphone device. For more information about the devices, see Section 4.3.

## 3.6 OUTPUT MODULES FOR DIALOG

The output modules which are created, are:

- **Text output** This module displays the output as text.
- **Speech output** This module outputs the given question as speech
- **Avatar output** This module outputs the given question as speech with an avatar

The avatar output module makes use of the Elckerlyc platform, which is a Behavior Markup Language (BML) realizer for real time generation of behaviors of virtual humans [16].

In Figure 8 [16] the architecture of the Elckerlyc platform can be seen. The Realizer Manager will send the question and possible answers to the Avatar output module, which will construct a correct BML string for the question. This BML string is send to the Elckerlyc platform, in Figure 8 represented as "BML stream".

The different engines will handle different parts of the BML stream and generate synchronized instructions for the specific parts, such as speech, body gestures, postures and facial expressions. This combined will be displayed by one embodiment such as a realistic 3D human animation or a 2D animation.

For our system we will use the PictureEngine for our mobile device. The PictureEngine is specifically made for the Android platform, it is lightweight - resulting in low battery usage.

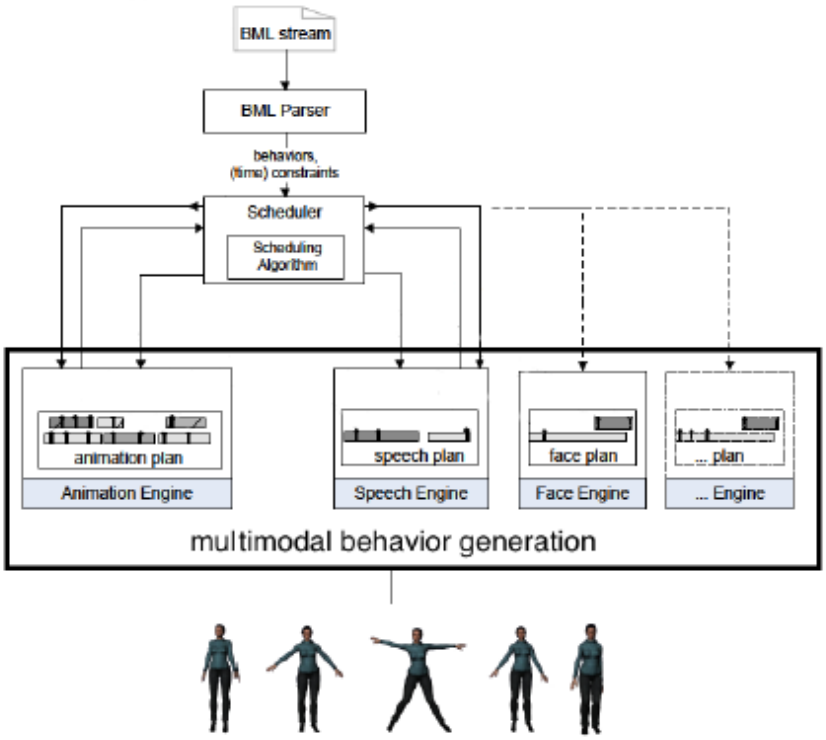


Figure 8: Elckerlyc Architecture

The desktop device will also run the PictureEngine, to keep consistency across the different devices.

Another reason, given by Klaassen et al., to not run the full body 3D engine on the smartphone, is that the screen of a smartphone is relatively small. This has as a result that the expressions given by the 3D avatar are hardly noticeable.

The only limitation of the PictureEngine is the lack of lip synchronization with the spoken text due to the text to speech engine used - which gives to little information about the utterances. The lips move but this is not based on the words being spoken. [16] This in contrast to the desktop version, where the lip synchronization works correctly.

Text output and avatar output will be supported by the desktop and smartphone device, speech output only by the smartphone device. For more information about the devices, see Section 4.3.

## 3.7 HEALTH DIALOG SYSTEMS

Health dialog systems are used more and more to support, provide information, advice and counsel patients [22]. One-on-one, face-to-face interaction with a health provider is, according to Bickmore et al., widely acknowledged to lead to the best health education and behavior change in patients. If it is possible to capture this in a health dialog system this will lead to better overall results, since software

can be replicated infinitely. This in contrast to health providers, such as doctors or psychologists. Thus a software system can reach much more people [6, 9].

Another advantage is, according to Bickmore et al., is that a dialog health system has unlimited time to spend with a patient - removing time pressure which normally could lead to patients being to afraid to ask questions or to ask for information to be repeated.

The second advantage is that a dialog health system always gives the data it has. This in contrast with human health providers, which can make errors or not follow recommended guidelines correctly.

The third advantage is that a patient may not always have access to all the health providers it needs, because of time, money or spatial issues.

Is there also something unique about health dialog itself, compared to other forms of dialog? Bickmore et al. give the following factors which can set health dialog apart:

- **Criticality** A dialog health system has the potential to be used in a emergency situation.
- **Privacy and security** The content of the dialog can be stigmatizing for the user, thus the applications should be sensitive the user's environment.
- **Continuity over multiple interactions** Communication about health is most of the times not a one-time thing, but require multiple interactions over extended periods of time - even a lifetime long when we think of applications which help with chronic diseases. The information retrieved in interactions can also influence the dialog which takes place at a later time, thus this information must be stored.
- **Language change over time** Normally when a patient interacts with a human health provider, the language adapts over time - becoming more concise and effective. Although this would be a nice feature for a dialog health system it is also necessary for a different reason. When an application always says the same thing, users lose interest which results in a less effective system.
- **Managing patterns of use** This is more a fundamental point about dialog health systems. How often should a user, or is a user allowed, to use the system. Is there an optimum for a given treatment?
- **Power, initiative, and negotiation** Again a fundamental point about dialog health systems, what role should such a system take? Should it be the professional or should it negotiate with its users?

- **User-computer relationship** The quality of the relation between a human health provider and the patient is a factor in patient satisfaction and treatment outcome. According to Bickmore et al. several studies have demonstrated that people respond to social cues from computers. When we combine this, a dialog health system should not only say the right thing at the right time, but it must also address social, emotional and relational issues in the dialog.

# 4 SYSTEM ARCHITECTURE

## 4.1 INTRODUCTION

In this chapter the architecture of the system will be described. What are we going to build and, in terms of the requirements, why is this necessary.

First we will start with an overview of the C3PO system, followed by the different devices we support in the system. After this section, the network setup of our system will be determined. Than the user interfaces will be shown and discussed. Followed by the dialog architecture and we finish with a total overview of the different devices and the interactions between them.

## 4.2 CONTINUOUS CARE & COACHING PLATFORM (C3PO)

The Continuous Care & Coaching Platform (C3PO), as described in Section 1.1.1, is developed in order to remotely and continuously monitor and treat elderly patients and patients with chronic disorders.

In the C3PO platform there is only one device with which the users interact, the smartphone. The users receive on the smartphone their feedback at scheduled times or if their activity level calls for this. The users can also fill in questionnaires.

In order to measure the activity, each user carries a ProMove-3D wireless sensor.

A C3PO application consists of a Hub and modules. Modules perform a certain functionality and can interact with other modules via the Hub. It is thus possible to easily create different applications to target different patients or contexts without recreating functionality.

In Figure 9 the architecture of a standard C3PO application can be seen. We will now describe the modules used in the system.

### 4.2.1 *Gui Module*

The Gui Module provides the Gui, as the name would suggest. Normally the activity graph is shown, with a button to show the latest feedback. It can receive messages from other modules to perform some kind of action on the screen. It can receive or request a feedback message from the Feedback Agent module or receive a questionnaire from the User Input module.

The activity graph uses data received from the IMA Data module.

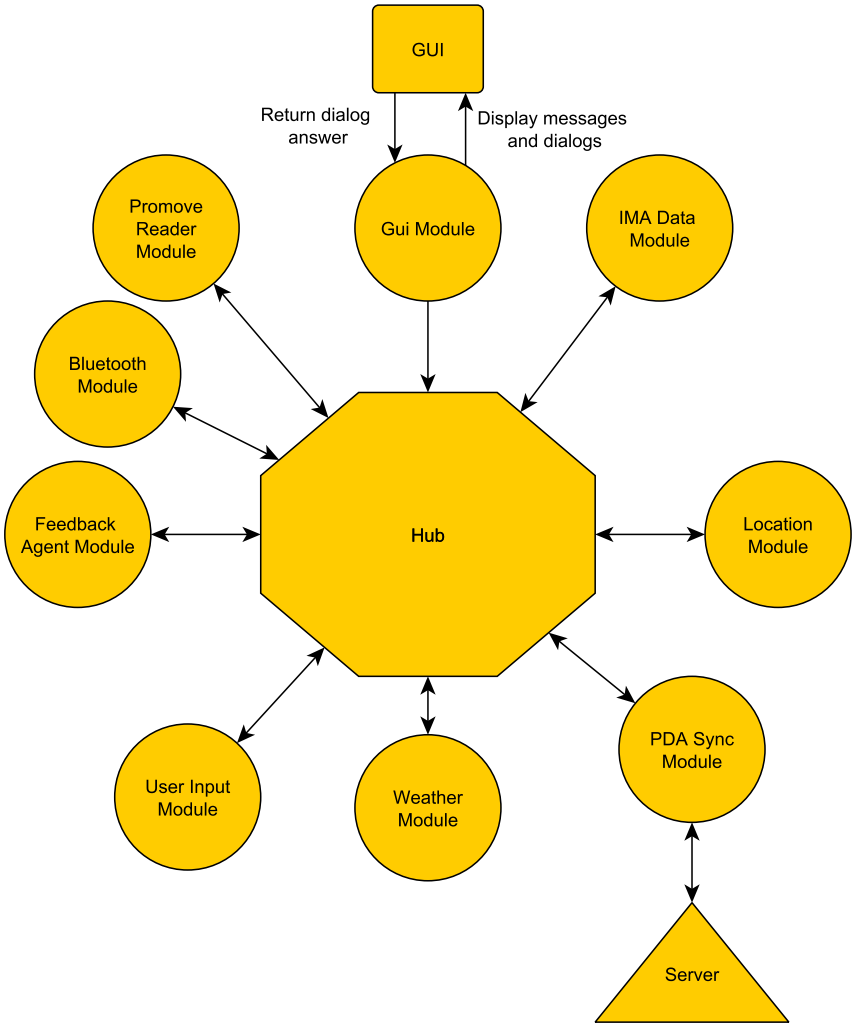


Figure 9: Architecture of a standard C3PO application

### 4.2.2 *Feedback Agent Module*

The Feedback Agent module can send feedback messages. The feedback type (encouraging, neutral or discouraging) of the message is based on the current performance of the user. The tone and content of the message is based on previous feedback messages and the reaction of the user to them.

The feedback agent remembers the messages it has send and after a certain time it evaluates if the message was a success, in other words the user did what the message said. This can be determined by looking at the activity of the user after when the message was send.

Some content of the message is also determined with the Weather module, if it is nice outside the feedback agent can suggest to go outside. Would the agent suggest to go outside while it is raining, the user will probably not follow the given suggestion.

### 4.2.3 *User Input Module*

The user input module reads the questionnaires and when they are scheduled. When a schedule trigger is hit, it sends questions to the GUI module. This is the dialog manager of the system and this will be adapted by us when necessary.

It stores the received answers.

### 4.2.4 *IMA Data Module*

The IMA Data module writes activity data to file and makes it also available to other modules, such as the Gui module.

It reads the activity data given by the Promove Reader Module.

### 4.2.5 *Promove Reader Module*

The Promove Reader Module reads the data from the Promove sensor, which has data from an accelerometer, gyroscope and magnetic compass.

### 4.2.6 *Bluetooth Module*

The Bluetooth module will enable a Bluetooth service on the smart phone and can connect to other Bluetooth devices, such as the promove sensor.

### 4.2.7 *Location Module*

The Location module tries to receive the location, as "<city>,<country>", of the user from the GPS provider or network provider.

### 4.2.8 *Weather Module*

The Weather module tries to receive the weather via the Google Weather API[1]. It uses the location received from the Location module and sends the weather info to the feedback agent module when necessary.

### 4.2.9 *PDA Sync Module*

This module can synchronize data between the server and the smartphone. Examples of data are the activity data of the user, user responses from the questionnaire or configuration files for the different modules.

## 4.3 DEVICES IN THE SYSTEM

This section will describe which devices we will support in the system. This section will describe what the different devices are and it will describe and show the Graphical User Interfaces (GUI) made.

It should be noted that is possible to run multiple smartphones, desktop (or laptop) devices and kiosks.

The capabilities of devices can be described with the following properties:

- **Mobility** Can the device be carried around? Devices such as a mobile phone or tablet have this capability.
- **Privacy** Is the device in a public space or not? A screen at a coffee corner is in a public space, the desktop computer is not. When a dialog is started which contains private questions these questions need to be skipped or the dialog itself should not be started at a public device.
- **Input Modality** Which types of modality supports the device for input? Input modalities are text input or speech input. Text input can be done via a keyboard, with a mouse an option can be selected.
- **Output Modality** Output modalities are text, speech or images. Also the size of the display. A dialog can require a minimum size of display, for example when there are a number of answers which should be shown to the user. Or a dialog has the preference to select the device with the largest display available.

1 http://www.google.com/ig/api?weather=<location>

- **User Attentiveness** Some types of dialog do not need a lot of attention of the user, such as non-important status updates. Dialogs with the user have a high attentiveness. Some devices also won't get a lot of attention of the user all the time, for example a display screen in a public space. We can't draw conclusions from low attentiveness messages, because we cannot be sure the user has seen it. In other words, messages from which the system does not want to learn anything have a low attentiveness score. Messages with a low attentiveness can be shown by multiple devices and if a device shows such a message it does not have to have the user object assigned.

### 4.3.1 *Smartphone*

The smartphone used for our system is a standard android based smartphone. The application running on the smartphone is also suitable to run on other android phones. We will only allow running the smartphone application on the smartphone, it is not possible to do other things besides it. This decision is made, because this is how it works in the normal C3PO platform.

We can define the device with the above mentioned properties in the following way:

| **Property** | **Description** |
|---|---|
| Mobility | Mobile |
| Privacy | Private |
| Input Modality | Touch, keyboard, text, speech |
| Output Modality | Small display, text, images, speech |
| User Attentiveness | High |

Table 9: Smartphone device properties

#### GUI

The smartphone application has several screens. In Figure 10 the home or default screen can be seen. This displays a graph with the user's activity performance and an advice button. With this device the last given feedback can be seen.

In Figure 11 the screen can be seen when the user receives a question from the questionnaire. The output realizer in this case is text.

### 4.3.2 *Desktop*

This is a normal computer, which everyone uses for his work or study. It can be a desktop computer or laptop and the application will run

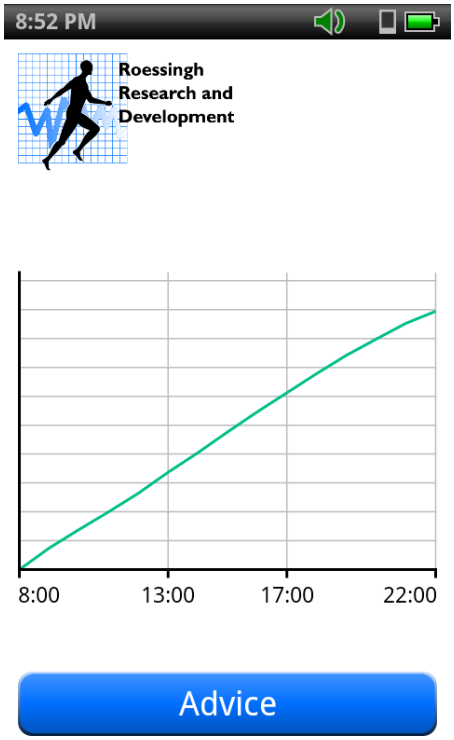


Figure 10: Smartphone home screen

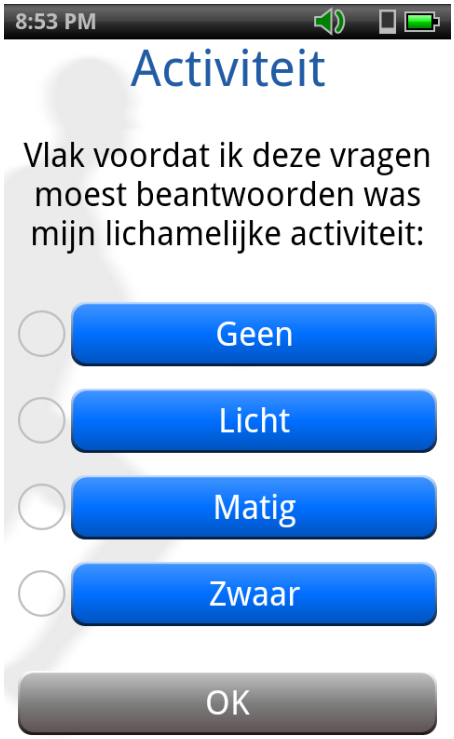


Figure 11: Smartphone Questionnaire Screen

on any computer as long as it has Java support. The user can do his normal work, without interrupting or interfering the application.

We can define the device with the above mentioned properties in the following way:

| Property | Description |
|---|---|
| Mobility | Not mobile |
| Privacy | Private |
| Input Modality | Mouse, keyboard, text. |
| Output Modality | Large display, text, images, speech |
| User Attentiveness | High |

Table 10: Desktop or laptop device properties

### GUI

Unlike the smartphone application set of screens, the desktop application's GUI consists of a taskbar icon and the different realizers for the input and output for the questionnaire questions and feedback messages.

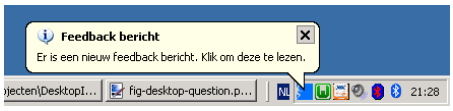


Figure 12: Desktop application taskbar icon and feedback reminder

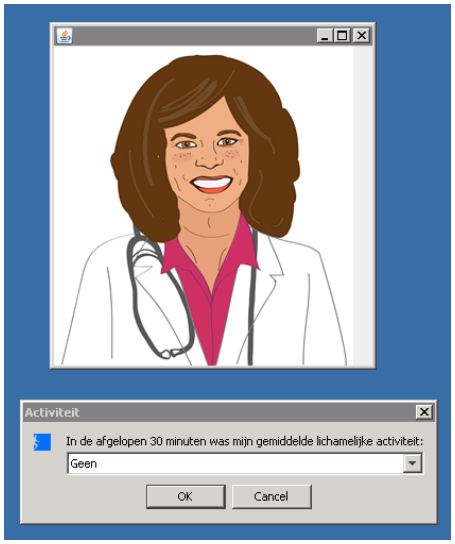


Figure 13: Question on the desktop application

In Figure 12 the taskbar icon can be seen on a Windows XP Operating System. Note that most of the icon is blue, this means the desktop

application has the user object. When another device has the user object, the icon is mostly gray - except the light gray if the running man. When the message is clicked, the feedback message will pop up or an avatar will be presented. This is determined by the selected realizer by the user.

In Figure 13 a question of the questionnaire can be seen, accompanied by an avatar which also asks the question. This again is determined by which realizer the user has selected.

### 4.3.3 *Kiosk*

The kiosk device is, seen from a user point of view, only a display. This can hang for example in a hall way or in the coffee corner. Reminders are displayed at the screen.

We can define the device with the above mentioned properties in the following way:

| **Property** | **Description** |
|---|---|
| Mobility | Not mobile |
| Privacy | Public |
| Input Modality | None |
| Output Modality | Large display, text, images |
| User Attentiveness | Low |

Table 11: Kiosk device properties

#### GUI


Figure 14: Questionnaire reminder on the kiosk

The kiosk has a simple GUI, it shows only a blue background or it shows some text to remind the user of the questionnaire when this is not yet finished. A smiley is added to lighten the mood. For an example see Figure 14.

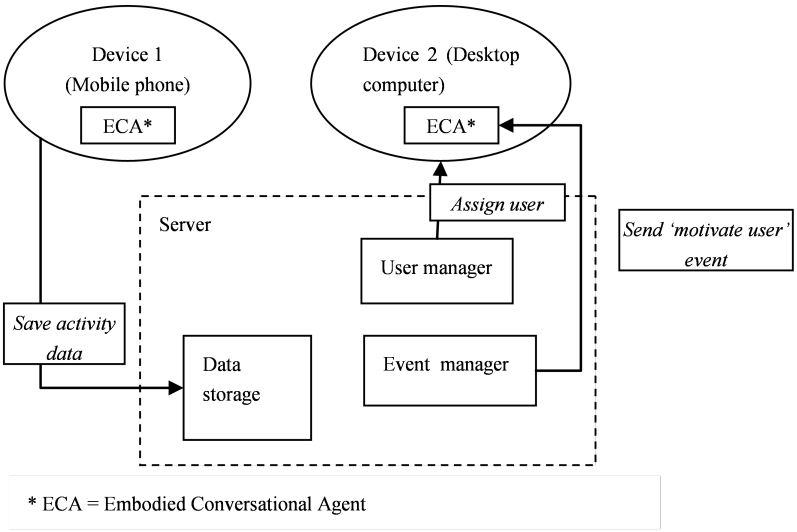


Figure 15: Central setup

## 4.4 NETWORK SETUP

This section will discuss with which kind of network we can develop our system in.

There are three possibilities for the setup of the system; central, decentral or hybrid. First we will explain the three different possibilities followed by their advantages and disadvantages. In the figures displaying the setups, the following situation is taken, the desktop computer has the user object assigned and should display a graph and motivating message.

### 4.4.1 *Central setup*

In the central setup, see Figure 15, there is one server where everything is saved and decided. At the server the data is stored, events are sent and the user object is assigned - all done by the respective components. All the events are sent by the event manager at the server to the device which interacts with the user.

### 4.4.2 *Decentral setup*

Another possibility is the decentral setup, which can be seen in Figure 16 and Figure 17. Here there is no central static place where everything is stored and decided. Everything is done in the mobile user agent itself, which can decide by himself to which device he goes. With mobile agent the definition given by [30] is meant:

> "[m]obile agents are autonomous programs that can travel from computer to computer under their own control"

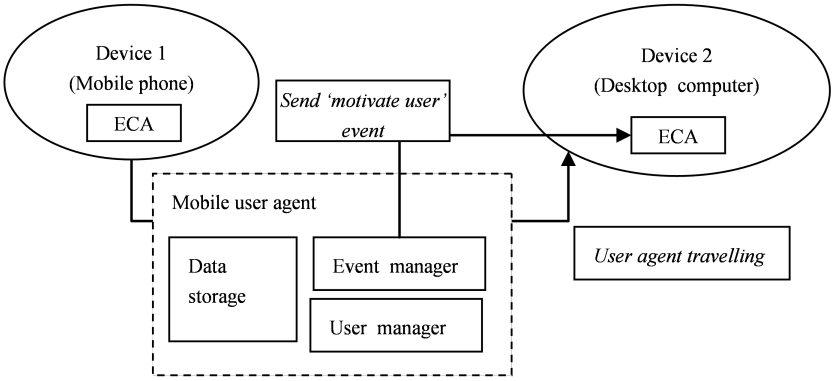


Figure 16: Decentral setup - version a

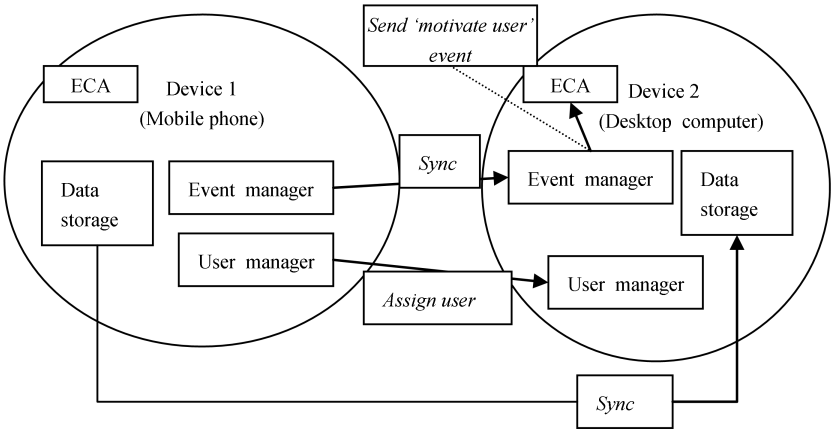


Figure 17: Decentral setup - version b

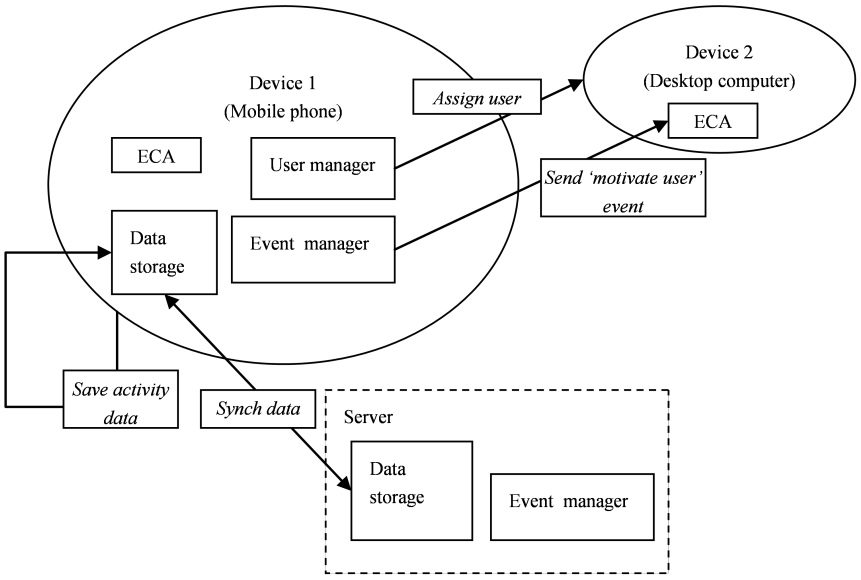


Figure 18: Hybrid setup

The decentral setup can also be done without using a mobile agent, see Figure 17. All devices have a data storage, event manager and user manager. In version a, the events are sent by the event manager of the mobile user agent to the device which has the mobile user agent. With version b, the events are sent by the event manager of the device which has the user. This is handled internally the event is handled by the proper component.

When the user manager of device A decides device B is better, all data and event data is synced with each other and the user object is assigned to device B. For this functionality it is necessary that each device must know where other devices are and what those other devices are capable off.

### 4.4.3 *Hybrid setup*

The third possibility is a combination of ideas of the central and decentral setup, the hybrid setup - see Figure 18. Here a device can have its own event manager and data storage. Or it can rely on a central server where he can save and retrieve data, as well as receiving events. The user manager should ideally be only on one device in order, as explained in the previous section, to avoid conflicts. When a device has its own data storage and event manager, the data and event rules need to be synchronized with the central server. When every independent device does this with the central server, each independent device also gets the data of the other independent devices. Events can be sent by each device which has an event manager to the device which interacts with the user.

### 4.4.4 *Evaluation*

The advantages for a central setup are that it is easier to create and maintain, because in this setup there are the fewest components. The version a decentral setup also has the same number of components, but a mobile user agent needs to be. A server, containing the components, in the central setup is needed as well, but this is such a standard setup multiple solutions exist for this. In the other setups there are the same number of component types, but these need to run on different kind of devices.

It is of course possible to make the components as modular as possible, but there needs to be a module interfacing with the operating system of the device.

The disadvantage for a central setup is when the central server goes down, nothing works anymore. Each device needs to use data and events from the central server. The system can also fail partially when a device loses connection with the central server. This means this device cannot interact with the user anymore.

When multiple data storages and event managers are running in the system, which happens in the decentral version b and the hybrid setup, data needs to be synchronized. In the decentral version b setup, data with the desktop only gets synchronized when a user passes with its mobile phone and the user manager assigns the user object to the desktop. When a device collects data it needs to wait for the user to be in the vicinity to get his data into the system. Or it needs to connect via an Internet connection to a data storage, for example the one in the mobile phone, to make his data known to the user - making the system more central.

Another point against the decentral and hybrid setups is how feasible it is, in terms of processing power and memory usage, to run the data storage and event manager on each device. When devices are not capable to run the data storage and event manager, they would be excluded in both versions of the decentral setup. Also the amount of data saved and used to have useful interactions with the ECA needs to be taken into account. This can be a problem when the amount of data is large that it is not possible to have this on, for example, a smart phone. In the hybrid setup there could be a possibility to have the latest data on the smart phone and older data is only available at the central server.

### 4.4.5 *Chosen setup*

For our system we will use the hybrid network setup. Each device has its own data storage and when the server goes down or there is no Internet connection, it is still possible for a device to operate (maybe in a limited way).

The devices we use for the start of our system can all run the necessary components by their own, in terms of battery and memory usage. When devices are added which are not capable to run this, they should rely on the central server.

## 4.5 DIALOG ARCHITECTURE

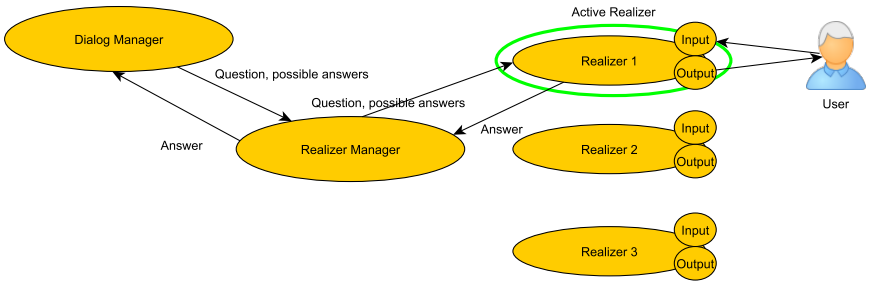


Figure 19: Dialog Architecture

In Figure 19 the overview of the dialog used in our system can be seen. The model underlying the dialog manager is in the definition of [6] an augmented transition network. The reason for this is that we want to simulate questionnaires, with the possibility to ask certain questions based on previous answers.

### 4.5.1 *Realizer Manager*

The Realizer Manager takes care of selecting a certain realizer and gives the received question and possible answers to the selected or active realizer.

A Realizer is a module which makes use of an input and output module. It directs the question to the output module and directs the possible answers to the input module. The input and output module are independent of each other, which makes it possible to combine the different in- and output modules into different realizers.

### 4.5.2 *Synchronization*

Our system is a Cross Media System in which a user handover between devices can take place. This means for the dialog system part that the previously given answers should be synchronized between all the devices. This makes it possible to continue a dialog at another device when a user handover takes place. The answers returned by the Realizer should also be picked up by the MDS Connection Module, see Figure 20. This module will send it to the Multi Device Server, which will distribute it between the other devices.

When a device receives a given answer, the local Dialog Manager should pick this up. The local Dialog Manager can than update its internal state accordingly. After a user handover the Dialog Manager then is already in the correct state and can continue the dialog where it was left off. This requires that each device has the same kind of dialogs at its disposal.

## 4.6 ARCHITECTURE OVERVIEW

In this section we will layout the general overview of the system, for each device and for the server. This will be followed by how the different devices and the server will interact with each other.

### 4.6.1 *Smartphone*

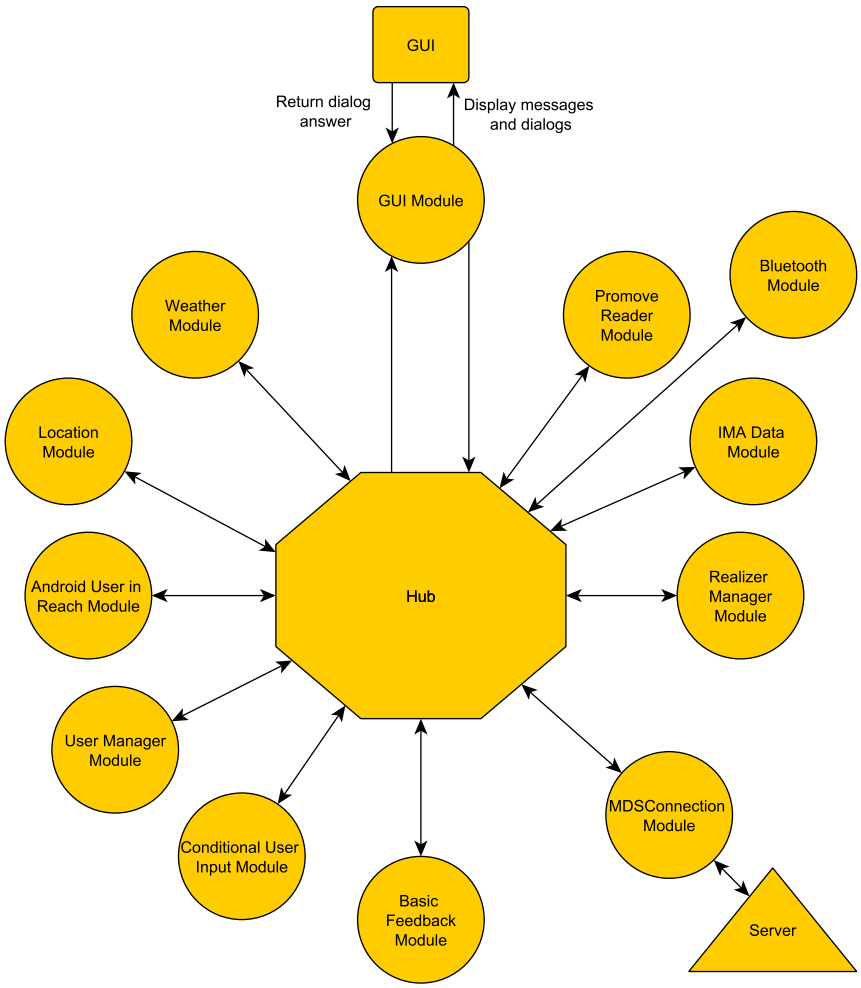


Figure 20: Architecture of the Smartphone Application

In Figure 20 the architecture of the smartphone application can be seen. We will now describe the different modules in the applications, omitting the (unmodified) modules mentioned in Section 4.2 - which discussed a standard architecture of C3PO.

### Conditional User Input Module

This module can be compared to the User Input Module mentioned in Section 4.2. The difference is that it will not propagate a questionnaire or feedback message when the smartphone does not have the user object.

This module also contains the dialog manager, it reads the questionnaire specifications available on the device and will schedule the questionnaire and feedback messages. When a questionnaire should start it will check if the device has the user object. If this is the case it will send one question to the Hub, this message will be received by the Realizer Manager Module.

When the question is send away, the module will eventually receive the answer given by the user. Depending on the questionnaire, the next question is selected or the questionnaire is ended. For more information about the dialog manager, see Section 4.5.

When this module receives a notification that the device has lost the user object, a cancellation message for feedback and questionnaire questions is send. This can be picked up by other modules and act accordingly.

If the received notification states that this device has been given the user object, the dialog manager will check if there is an open questionnaire. If this is the case the question is send to the Hub, after which other modules can act on it.

### Realizer Manager Module

When the device is started, modules can register their realizers with this module. In the current case this is done by the GUIModule.

This module is subscribed to questions from the dialog manager / conditional user input module. When it receives such a question it will determine which realizer to chose. If the user has given a preference it will select those input and output realizers. If this is not the case than it will select the most rich realizers available. Where the avatar realizers is the richest, the speech realizer follows the avatar realizer in rank and the text realizer is the lowest on the scale.

When the system is continued further, this module is a good place to make a device context aware.

### GUI Module

The GUI module has been adapted from the module as used in Section 4.2. It contains the input and output realizers described in Section 4.5.1.

During a questionnaire or with a feedback message, the user has the possibility to change his preference for the input and output realizers. When this preference is changed, the current question or feedback message is repeated in the preferred realizer. After the update new questions or feedback message will be done with the preferred realizers, because the Realizer Manager module is notified of this change.

The ability to switch the modality is recommended by Norman ([24], as cited by [27]), who states that allowing the user to switch modalities when he wants to may ease interaction when the user interacts with a mobile device. When the user uses the mobile device when he walks, his visual attention is focused on his primary task - walking. The smartphone now can only request minimal attention and switching to other modalities may improve the interaction in that situation [27].

During a questionnaire the answer of the user is send to the Hub, which notifies other subscribed modules.

When the smartphone releases the user object, the GUI Module is instructed to cancel all interaction.

### Android User in Range Module

This module determines if the user is in range of the module and the user is always in range of the smartphone - this is an assumption we made, see also Section 2.6.5. We always need a device on which we can display information - see Section 2.4.

Because of this assumption and requirement, this module will only broadcast a messages which states that the user is in range at the start up of the application.

### User Manager Module

This module requests the user object when the user is in reach, as described in the previous paragraph, this will be done at the startup of the application. When the desktop application requests the user object, a handover takes place. When the desktop application releases the user object, the server will always hand the user object over to the smartphone. That is why it is not necessary to request the user object again by this module for the smartphone application.

Would this application request the user object from the server and a desktop application has the user object, its request would be denied.

### MDS Connection Module

This module takes care of the connection with the server. It will pass user request messages to the server, as well as answers given by the user to questionnaires.

The module also passes messages received from the server to the Hub. Messages pertaining to user handover (receiving or losing the user object) and of answers of questionnaires given by the user on other devices. The question answer for example is received by the Dialog Manager / Conditional user Input module, which will move the state of the dialog forwards. This in turn leads to the fact that the dialog on each device is in the same state after an answer is given by the user.

## 4.6.2 *Desktop*

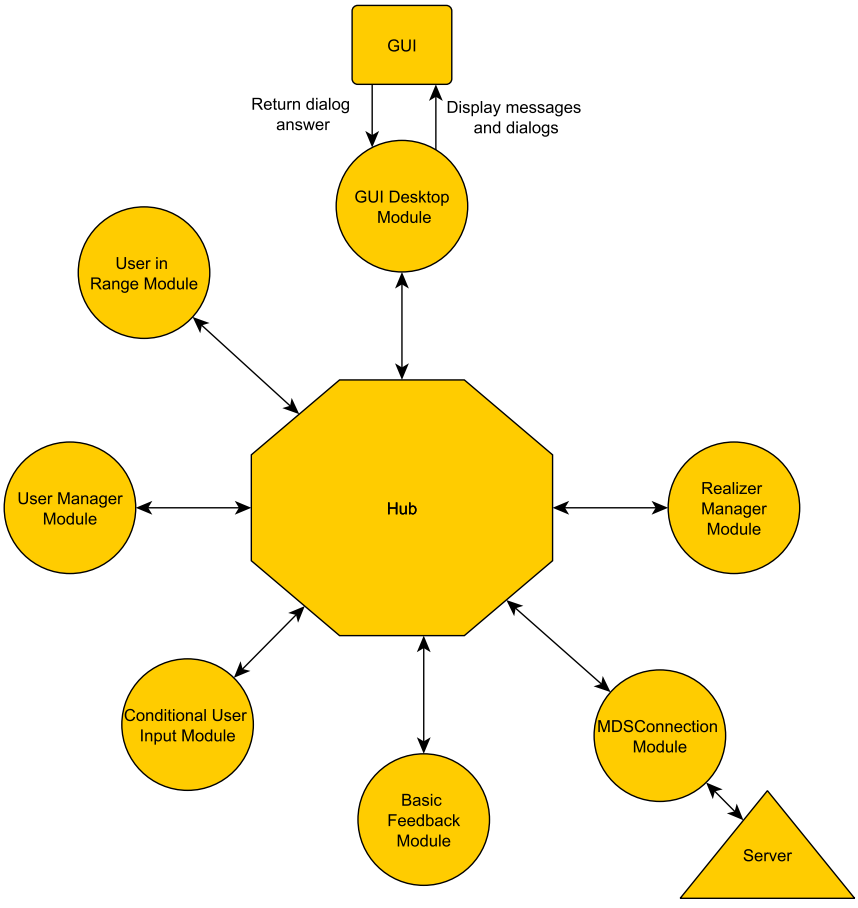


Figure 21: Architecture of the Desktop Application

In Figure 21 the architecture of the desktop application can be seen. We will now describe the different modules in the applications,

omitting the (unmodified) modules mentioned in Section 4.2 and Section 4.6.1.

#### User in Range Module

This module will send a message that the user is in range when the user uses the mouse or keyboard. It will send an out of range message when no mouse or keyboard activity is detected for thirty seconds.

#### GUI Desktop Module

This module takes care of the GUI on a desktop (or laptop) computer. It displays an icon in the taskbar which shows if the device has the user object (mostly blue icon) or not (mostly gray icon).

When a feedback message is received it will display a reminder balloon, indicating the user a feedback message is available. If the user clicks this balloon the feedback message is shown.

Questionnaires are also displayed by this module and it supports text and avatar output realizers and text input realizers.

By right clicking the taskbar icon, a menu pops up. Via this menu, the user can close the application or switch his preferred realizers. There is also the option to show a frame containing the server and hub messages received in the current session, although this is only useful for developers.

### 4.6.3 *Kiosk*

In Figure 22 the architecture of the kiosk application can be seen. We will now describe the different modules in the applications, omitting the (unmodified) modules mentioned in Section 4.2, Section 4.6.1 and Section 4.6.2.

#### User in Range Module

Unfortunately the functionality for determining a user being in range with bluetooth is not yet finished. That's why this uses the same module as the desktop application.

#### GUI Kiosk Module

This module shows a blue screen. When there is a user in range and a questionnaire is not completed, a reminder is shown on the screen.

### 4.6.4 *Server*

In Figure 23 the architecture of the server can be seen. We will now describe the different components of the server.

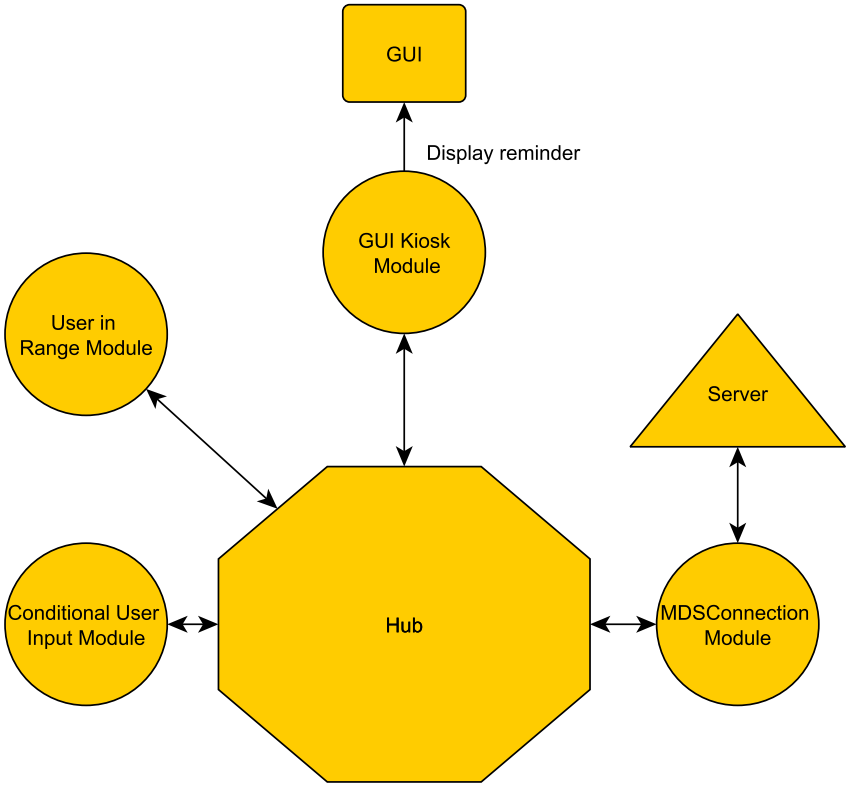


Figure 22: Architecture of the Kiosk Application

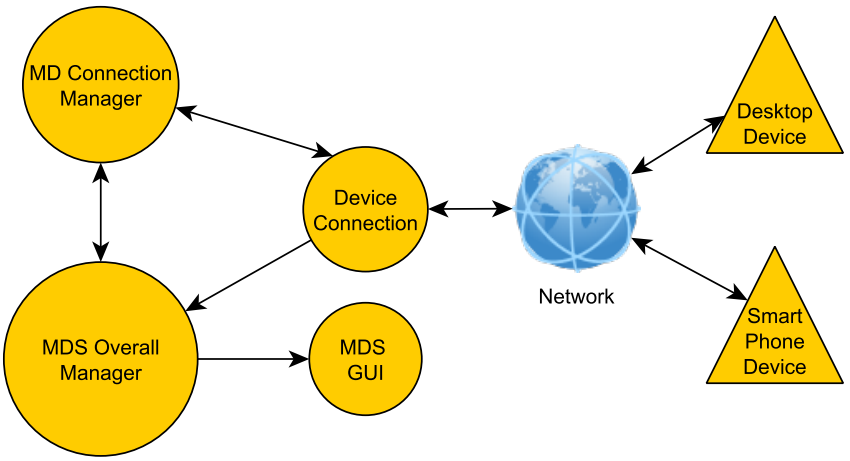


Figure 23: Architecture of the Server

#### MD Connection Manager

The MD connection manager takes care of the communication between the devices registered at the server and the MDS Overall Manager. When a device registers itself with the server it gets a connection manager assigned. When the connection manager receives a request for a user handover, it asks the overall manager if this is allowed.

When a device releases the user object, the connection manager informs the overall manager that this has happened.

If a device sends a questionnaire answer it will notify the overall manager which in turn will notify the other connection managers.

When a user handover takes place and the device of the current connection manager loses the user object, the connection manager will inform the device.

#### MDS Overall Manager

The previous paragraph contains already much information about the MDS overall manager. This is the place in the server where all the decisions are made, every device is registered and it also sends relevant information to the server gui.

#### MDS GUI

This shows relevant messages in a GUI and the server can be stopped by closing the GUI.

### 4.6.5 *Interaction*

In this section several interactions between devices and server are shown.

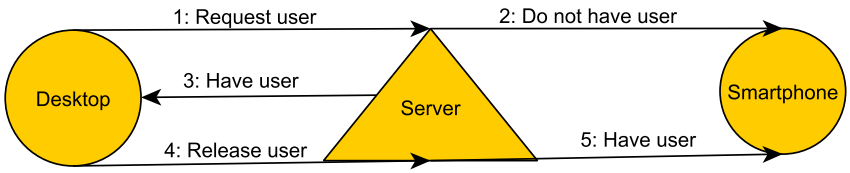


Figure 24: User Handover interaction

In Figure 24 the interaction with two user handovers can be seen. At the start of this interaction the smartphone has the user object. At point 1 the desktop requests the user object, because the user is in range of the desktop. The server informs the smartphone at point 2 that it does not have the user object anymore. At point 3 the desktop is informed that it has the user object.

At point 4 the desktop detects that the user is not anymore in range and releases the user object. The server informs the smartphone at point 5 that it has the user object.

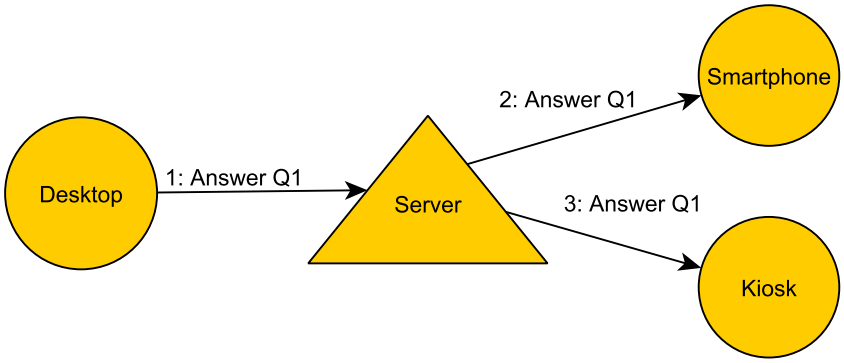


Figure 25: Answer Synchronization Interaction

In Figure 25 the interaction with answer synchronization can be seen. At each device the questionnaire is started, but only at the desktop device is the question presented to the user. This means also that the question can only be answered at the desktop. At point 1 the user gives the answer, which the desktop sends to the server. At point 2 and 3 the server sends the answer to the smartphone and kiosk device.

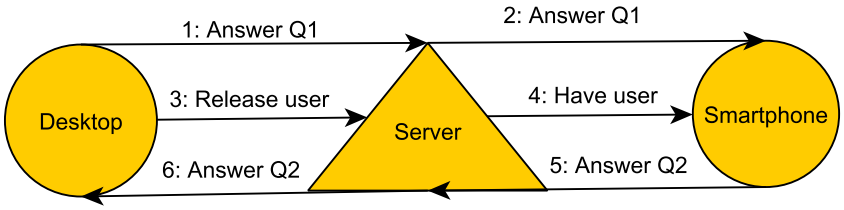


Figure 26: Answer Sychronisation and User Handover Interaction

In Figure 26 the interaction can be seen when during a questionnaire a user handover takes place. At point 1 the desktop presents the question to the user, who answers it. This answer is send to the server and at point 2 the server sends the given answer to the smartphone.

Apparently the user does not answer the next question and does nothing anymore on the desktop. At point 3 the desktop releases the user object and the server informs at point 4 that the smartphone has the user object.

The smartphone detects that there is an unfinished questionnaire and presents the current question to the user. The user answers this question and at point 5 the smartphone sends the question to the server. The server sends the given answer at point 6 to the desktop.

# 5 SYSTEM IMPLEMENTATION

## 5.1 INTRODUCTION

In this chapter the implementation for the architecture of the system will be described. First the smartphone application will be described, followed by the desktop and kiosk application. This chapter will be finished with sections on the server implementation and dialog implementation.

## 5.2 SMARTPHONE APPLICATION

Each application is build with modules. A module does a specific thing and several modules combined result in a working application. Modules can interact with each other via the Hub, through messages. The Hub is the location where the modules are started and through which the messages pass. Modules can register in what kind of messages they are interested and they can act on them. They can use the message to perform a certain function themselves or if the message is a request, to send the information requested to that specific module (via the Hub).

In Figure 27 the setup for a device can be seen. We have limited this overview with modules pertaining to the cross media and dialog functionality. Functionality omitted is about the activity readings and the weather and location modules, since the focus of the thesis are the previously mentioned functionality. Also the modules pertaining to the activity readings and the weather and location modules were already implemented by Roessingh Research and Development

### 5.2.1 *GUI Module*

When this module is started it registers the available realizers it has with a MDS_DIALOG_REALIZER message, this contains the id of each input and output realizer this module has.

The GUI Module listens to FEEDBACK_DATA, MDS_DIALOG_ASK_QUESTION and GUI_CANCEL_USER_INPUT messages. When it receives a FEEDBACK_DATA message a new screen is presented to the user which contains the feedback message.

If a MDS_DIALOG_ASK_QUESTION message is received a user input event is started. The MDS_DIALOG_ASK_QUESTION contains the question string, title string, answer strings, answer ids, input realizer id

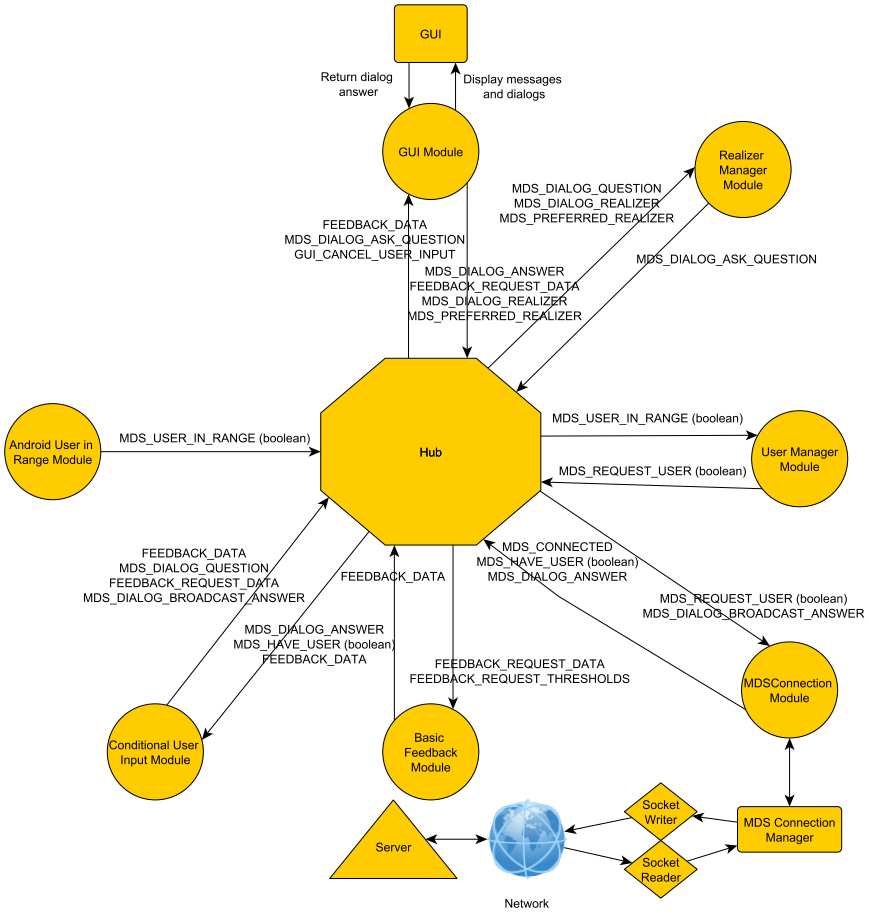


Figure 27: Implementation of the Smartphone Application

and output realizer id. The event is then executed and a new screen is presented to the user.

When the input realizer is speech, the Android Speech to Text engine is started. If the output realizer uses sound, the Speech to Text engine is started after the last output utterance is completed - this to prevent input detection based on the spoken output.

The Android Speech to Text engine records sound and sends this to the Google service. The language of the sound is set to Dutch which is also send to the Google service.

The results received from the Google service is a list of strings and each entry is a guess by Google what could be said. The first entry has the highest confidence and each following entry has a lower confidence.

The list with all of the entries, and each entry can contain multiple words are flattened. With this is meant that we create a list which contains per entry one word which Google probably thinks is uttered. Then this list is checked if it contains the answer string, if this is the case we assume that that answer has been chosen by the user.

If that is not the case, we will look for numbers and if a number is found we treat this as an index number for the answer list. If an answer is found we assume the user meant this answer. This makes it possible for the user to say "Antwoord twee".

If an answer is already selected we will also check if the flattened utterance list contains "ok", "oke", "okay", "ga" AND "verder" or "volgende" AND "vraag". When this is the case the previously selected answer is send to the Hub in a mds_dialog_answer message.

When the Google service reports an error, for example it cannot recognize anything, the Speech to Text action is started again. This also happens when an answer has been given, but not an okay command.

When the input realizer is text, the user can select the answer from a list. This is also possible when the input realizer is speech, since probably most of the times the user also wants to have the opportunity to enter it by hand.

If the output realizer is text only text is outputted for the question. When this is speech, the Android Text to Speech engine is started and the question string is passed along.

When the output realizer is the avatar, the Android Text to Speech engine is started for the speech and an activity for the avatar is started. This activity contains BML and because our BML is quite simple we can splice the question string into it.

The user also has the possibility to change the used input and output realizers. This can be done during a question and feedback message. These options can be found when the user scrolls down and via radio buttons can set the preferred realizers.

When this occurs a MDS_PREFERRED_REALIZER message is send, which contains if it is the input or output realizer and the specific realizer

id. The question or feedback message is repeated with the preferred realizer when the output realizer is updated.

If this module receives a GUI_CANCEL_USER_INPUT message, running questions or feedback messages are cleared from the screen.

### 5.2.2 *User Manager Module*

This module takes care of requesting the user when applicable with the MDS_USER_IN_RANGE message. In the case of the smartphone application this is done always when a MDS_USER_IN_RANGE message is received, which is only when the application is started.

### 5.2.3 *MDS Connection Module*

The MDS Connection module maintains the connection with the Multi Device Server via a TCP socket. It runs the socket reader and socket writer in different threads, making it possible to send and receive information simultaneously.

When it is connected with the server it sends a MDS_CONNECTED message. It listens for the MDS_REQUEST_USER and MDS_DIALOG_BROADCAST_MESSAGE and send these to the server. It receives the MDS_HAVE_USER and MDS_DIALOG_ANSWER from the server and will forward these to the Hub.

When this module is started, it reads the server adres and socket from a configuration file.

### 5.2.4 *Basic Feedback Module*

This module is already available in the PDA Platform and sends FEEDBACK_DATA messages. It sends a message when it receives a FEEDBACK_REQUEST_DATA.

The feedback messages are defined in an xml file and based on the current activity deviation point it is determined which type of feedback is send; encouraging, neutral or discouraging.

### 5.2.5 *Conditional User Input Module*

The Conditional User Input module is based on the User Input module which is normally used in C3PO.

When this module is started it reads from a configuration file the different schedules, which contains questionnaires and/or feedback messages. See for more information Section 5.6.

If a questionnaire should be started and the device has the user object, it will send the first question with a MDS_DIALOG_MESSAGE. When it receives a MDS_DIALOG_ANSWER message it will determine if

it needs to stop the questionnaire or - based on the answer - select the next question.

Also when it receives the answer, it will send a MDS_DIALOG_BROADCAST_ANSWER which indicates it needs to be synced with the server.

When a feedback message is scheduled and the device has the user object, it will send a FEEDBACK_REQUEST_DATA message to the basic feedback module, with the instruction to only send the feedback_data message back to the conditional user input module. After the FEEDBACK_DATA message is received it sends a global FEEDBACK_DATA message and the modules registered on this Hub message can for example display the feedback.

### 5.2.6 *Android User in Range Module*

This module will send a MDS_USER_IN_RANGE message with the value true, when this module is started.

## 5.3 DESKTOP APPLICATION

In Figure 28 the implementation of the desktop can be seen, it does not differ so much when compared to the smartphone application. We will now only treat modules which have not been mentioned in the previous section or if there is some change in functionality because of different messages received.

### 5.3.1 *GUI Desktop Module*

When this module is started it registers the available realizers it has with a MDS_DIALOG_REALIZER message, this contains the id of each input and output realizer this module has. It also places an icon in the taskbar if the operating system where the application is being run on has support for a taskbar.

As with the GUI Module of the smartphone application, the GUI Desktop Module listens to FEEDBACK_DATA, MDS_DIALOG_ASK_QUESTION and GUI_CANCEL_USER_INPUT messages.

The only difference with the GUI Module is that it does not support the speech input realizer. The text input realizer is started from the Desktop GUI module.

The avatar output realizer in the GUI Desktop Module is started via an ActiveMQ message. The actual avatar realizer on the desktop is a stand alone program, which also registers itself to the ActiveMQ service. The ActiveMQ message is passed by the ActiveMQ service to the avatar realizer. For more information on ActiveMQ see Section 1.1.1.

As with the GUI Module the given answer of the user is send to the Hub.

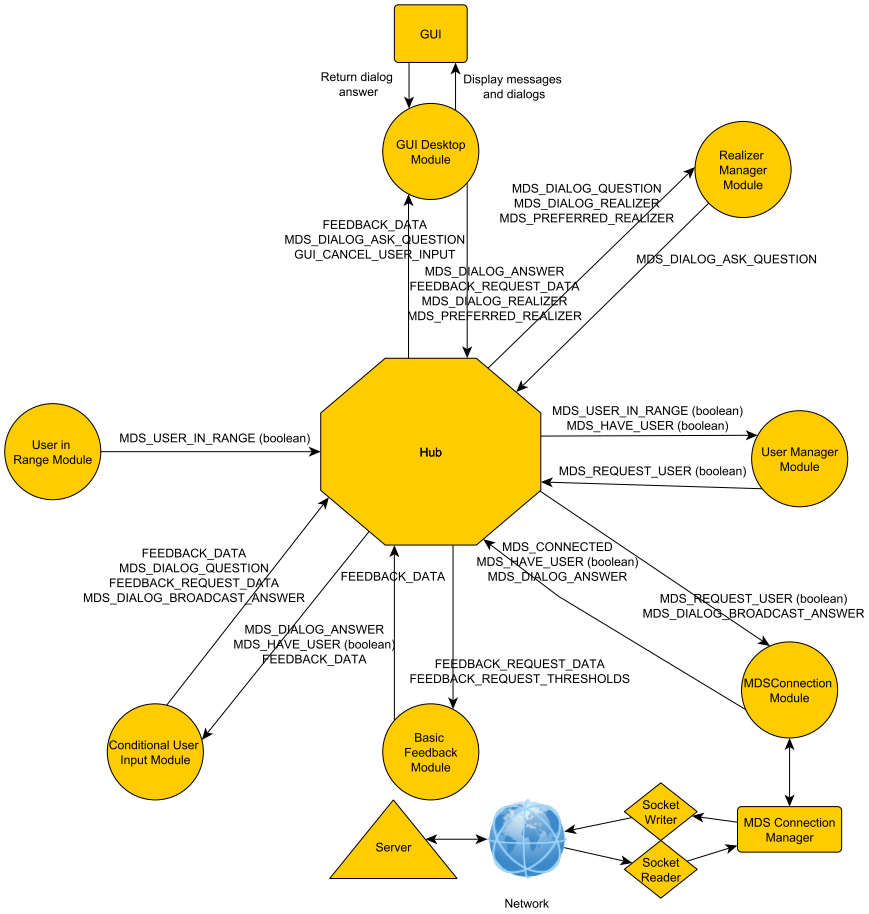


Figure 28: Implementation of the Desktop Application

### 5.3.2 *User in Range Module*

This module will detect if the user is in range by monitoring keyboard and mouse input. It will load a Windows dll via JNA, the Java Native Access library[1][2].

A thread is started at the start of this module which checks every second how long it was when the user has last used the mouse or keyboard. If this is shorter than thirty seconds and before the current check the user was out of range a MDS_USER_IN_RANGE message is send with the value true.

If this value is higher than thirty seconds and before the current check the user was in range a MDS_USER_IN_RANGE message is send with the value false.

### 5.3.3 *User Manager Module*

This is the same User Manager Module as in the smartphone application only now it receives more MDS_USER_IN_RANGE messages. If that message has the value true and the device does not have the user object, it will send a MDS_REQUEST_USER message with the value true.

If the MDS_USER_IN_RANGE is false and the device has the user object it will send a MDS_REQUEST_USER message with the value false, in effect releasing the user object.

## 5.4 KIOSK APPLICATION

In Figure 29 the implementation of the kiosk application can be seen and only the relevant messages are shown. The function of the kiosk is to show a reminder when the user is in range and there is an unfinished questionnaire.

This functionality is implemented in the following way. When a user is in range, the MDS_USER_IN_RANGE with the value true is send by the User in Range Module. The GUI Kiosk module is registered to this message and will than send a MDS_REQUEST_OPEN_DIALOG, indicating it wants to know if there is an open questionnaire.

The Conditional User Input Module will respond with a MDS_OPEN_DIALOG message with the value true. When the GUI Kiosk module receive this message - and the user is still in range - it will display the reminder.

The Conditional User Input Module will know the status of the questionnaire, because it receives the MDS_DIALOG_ANSWER message from the server via the MDS Connection module.

1 https://github.com/twall/jna

2 The code is based on the website "Detect the user's inactivity in Java with JNA", which can be found at http://ochafik.com/blog/?p=98

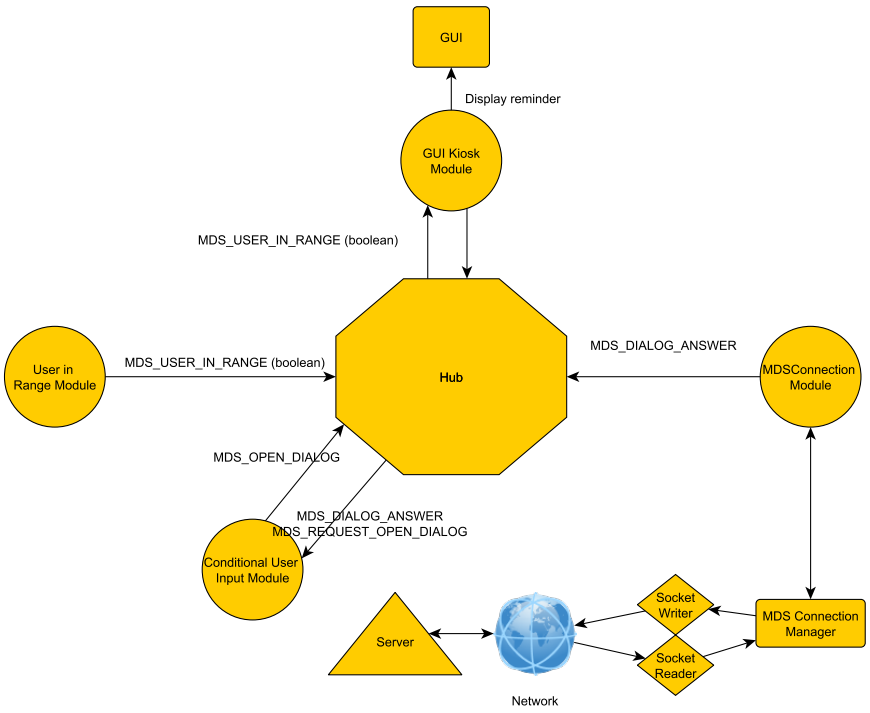


Figure 29: Implementation of the Kiosk Application

## 5.5 MULTI DEVICE SERVER

The Multi Device Server keeps track of all the devices, stores the user data and sends messages to the user via a selected device. In Figure 30 the setup for the server can be seen.

A device is seen as active when it has a connection with the server, later on this can be adapted in sending status messages back and forth in order to see if a device is available. This is currently not necessary, because a stable Internet connection is assumed.

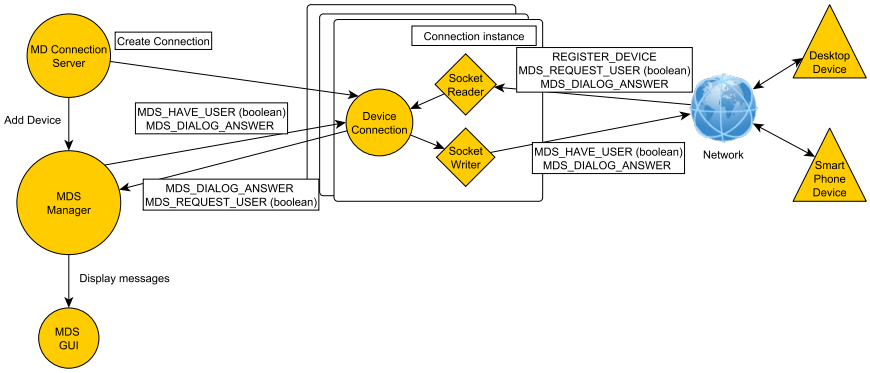


Figure 30: Multi Device Server

A simple GUI is available which shows connected devices and messages send and received. It is not necessary to run the GUI when running the server.

When a smartphone register itself it will check if a device has the user object, if this is not the case the smartphone will receive the user object after registration.

When it receives a MDS_REQUEST_USER of a device it will check if a device has the user object, if not the requesting device will receive the user object. If the device which has the user object is the smartphone and the requesting device is a desktop application, the requesting device will receive the user object.

If a requesting device does not receive the user object the message MDS_HAVE_USER with the value false is sent to that device. When a device receives the user object, that device will receive the message MDS_HAVE_USER with the value true. All the other devices will receive the message MDS_HAVE_USER with the value false.

When the server receives a message MDS_REQUEST_USER false from the device which has the user object, it will send MDS_HAVE_USER false back and also clear internally that this device has the user object. It will then by default give the user object to the smartphone, sending this device the message MDS_HAVE_USER true and to the other devices MDS_HAVE_USER false.

When the server receives a MDS_DIALOG_ANSWER from a device, it will send the MDS_DIALOG_ANSWER to all the other devices.

## 5.6 DIALOG REPRESENTATION

In the following section it will be explained how the dialog is represented in the Dialog Manager. Each specific dialog is in its own file, which is an XML file. The dialog representation used is based on the one used in C3PO.

Listing 1: Strings Example

```
<strings locale="nl_NL">
      <string id="activity_level_30min_question" value="
          Mijn activiteiten niveau in the laatste dertig
          minuten was:" />
      <string id="activity_level" value="Activiteiten
          niveau" />
      <string id="none" value="Geen" />
      <string id="mild" value="Laag" />
      <string id="moderate" value="Gemiddeld" />
      <string id="heavy" value="Hoog" />
</strings>
```

Each file should contain a strings section, which can be seen in Listing 1. The locale is for which country and language the values are. The id attribute is the id of the string, which is used to reference that specific string in other elements. The value is the string value.

Listing 2: Point Schedule Example

```
<schedule
      type="point"
      start_date="2010-01-01"
      end_date="2011-01-01"
      recur_date="1 day"
      start_time="08:00:00"
      end_time="20:00:01"
      recur_time="2 hours"
>
      ...
</schedule>
```

Listing 3: Interval Schedule Example

```
<schedule
      type="interval"
      start_date="2010-01-01"
      end_date="2011-01-01"
      recur_date="1 day"
      start_time="18:00:00"
      duration="16 hours"
      notify_time="20:00:00"
>
      ...
</schedule>
```

Listing 4: Continuous Schedule Example

```
<schedule
      type="continuous"
      pause="5 seconds"
      >
      ...
</schedule>
```

The schedule element specifies when a dialog should be started with the user. The schedule type can be one of the following three types:

- **Point** A point schedule will exactly start a dialog at the given point in time. It can be repeated several times. An example can be seen in Listing 2.

- **Interval** An interval schedule will start a dialog at a certain interval at can be done only once in that interval. It is possible to repeat the request on multiple days, but only one interval per day is allowed. An example can be seen in Listing 3.
- **Continuous** A continuous schedule will start a dialog and repeat the dialog after the given pause after completion of the dialog. An example can be seen in Listing 4

An element schedule will contain one element request and the request element will contain (multiple) input elements. The element input should contain one element, which should be feedback, multiple choice or open question.

Listing 5: Feedback Example

```
<feedback module_id="mod_feedback" method="interval"
    interval="1 hour" />
```

In Listing 5 a feedback example can be seen. It is defined with the following attributes:

- **module_id** If there are multiple feedback modules in the system, this specifies the feedback module.
- **method** The method how the activity level of the user should be calculated. If it is interval, the deviation line is calculated for the given interval, if it is day the deviation line for the whole day is calculated. If it is integrated, both will be used.
- **Interval** The interval period.

Listing 6: Example of Multiple Choice Question

```
<input id="activity_level_30min">
      <multiple_choice
            title="activity_level"
            question="activity_level_30min_question"
      >
            <answer text="none" value="none" next="feedback1
                "/>
            <answer text="mild" value="mild" />
            <answer text="moderate" value="moderate" />
            <answer text="heavy" value="heavy" next="NONE"/>
      </multiple_choice>
</input>
```

In Listing 6 an example of a multiple choice question is given and it consists of a question and a list of possible answers. The attributes text, title and question all refer to a string id - which are specified in

the strings section. The value attribute is the answer id used to denote the answer.

The next attribute is optional, but if it is given it refers to an input id. This will be the next step in the dialog if that answer has been chosen. The attribute can also contain the value NONE, which means the dialog will end. If no next attribute is given, the next input after the current input will be the next one in the XML file.

Listing 7: Example of an Open Question

```
<input id="question_sit_ups">
      <open
            title="situps"
            question="situps_question"
            type="value"
      />
</input>
```

In Listing 7 an example of an open question can be seen. The title and question attributes refer to a string id. The type attribute can be value or string. When it is a value a number is expected to be given, when it is a string it can be any text.

# 6 EVALUATION

In this chapter we discuss how we evaluate our prototype with users. In Section 6.1 we describe what and how we did the evaluation. In Section 6.2 we present the results of the evaluation.

## 6.1 METHOD

The evaluation consists of usability testing and the "Thinking Aloud" procedure, this combined with a testing scenario the test participant must perform. What we tested is if our developed prototype handles the use cases from Section 2.2 correctly and if our design decisions are correct.

### 6.1.1 *"Thinking Aloud" Procedure*

The "Thinking Aloud" procedure, sometimes also called "Thinking out loud", is a technique where the user verbalizes his thoughts [5]. The test participant will express his thought process by saying what he thinks during a certain task [29]. Rubin et al. [29] state that this technique is particular effective in early exploratory research - for example prototype testing in our case - and to get the participant's mental model of the application.

Van den Haak et al. [35] state that there are two versions of the "Thinking Aloud" technique, concurrent thinking aloud and retrospective thinking aloud. With the first version the participants think aloud during the test and with retrospective thinking aloud the participants are filmed and afterwards they comment on what and why they did something.

Barnum states that it is possible that the response time of the participants goes up and thus this technique should not be used for timed tasks [5]. With retrospective thinking aloud the response time is not affected [35]. The disadvantage of retrospective thinking aloud, according to Van den Haak et al., is that participant may give biased accounts of the thoughts they had while performing the task. They could have forgotten certain steps. Another disadvantage of retrospective thinking aloud is that it is considerably longer than the concurrent version.

Rubin et al. say that some participants might find concurrent thinking aloud unnatural and distracting and when it is done for a long time it gets exhausting. Although Van den Haak et al. [35] found in

their research that on average the participants - both in concurrent and retrospective thinking aloud - rated the experience neutrally.

For our evaluation we will use the concurrent thinking aloud technique, because it is shorter and people will not forget what they did when they say their thoughts at the moment they do a certain task.

Barnum [5] gives the following techniques to encourage thinking aloud:

- **Prompting** Focus on tasks, not on features. And focus on questions, not on answers.
- **Echoing** Repeat the participants words or phrase back as a question.
- **Conversational disequilibrium** Do not finish your statements or questions, but let the participants finish them. Indicate that it is their turn to talk.
- **Summarizing at key junctions** When you have learned something new, summarize the event and thinking of the participant briefly. The participants may offer more detail. And at the end of the session, leave the recorder on. The participants often make interesting reflections about the process at the end of the session.

### 6.1.2 *Usability Testing*

Rubin et al. [29] state that usability testing is a process where the test participants are members of the target audience. The product is tested to evaluate the degree in which it meets specific usability criteria.

What are our usability criteria? These usability criteria's were already mentioned in Section 1.2, but we will repeat them here.

The system we are going to test is a cross media system and Oquist et al. [27] give the following usability factors which are of importance for a cross media system:

- **Portability** How portable is the device which should be carried around. In our case it is the smartphone, although many different smartphones can be used in our system we can still use one specific smart phone and see how easily people carry it around when they are leaving for example their workplace.
- **Attentiveness** Oquist et al. state that when portability of a device increases, the amount of attention it can demand goes down. An example in our case would be if users would notify that they receive a questionnaire when they are walking and how easy is it for them to fill in that questionnaire. Does it help the user that they can switch the modality or realizers when they receive a questionnaire?

- **Manageability** How easily can users manage the GUI of the different devices. In the case with the smartphone, can the user use only one hand to interact with the application.
- **Learnability** How easily can the user understand and use the different functionality and when switching between devices does the user experience it as the same interaction paradigm.

More usability criteria for cross media systems, or multi device systems, are given by Denis et al [10]. They coin the term inter-usability and with this they mean how easily users can reuse their knowledge and skills for an action when using another device (see also Section 2.4). Inter-usability has two dimensions [10]:

- **Knowledge Continuity** How easily can users understand how to perform certain functions supported by the system with the different devices. Ideally this is the same for each device, although Denis et al. acknowledge this is not feasible, given the constraints of mobile devices. Knowledge continuity usability points to look at are:
  - **Visual appearance and continuity** Can the user find the same service in the same place and does he think the interface look the same. For example, have buttons on different devices with similar functionality the same label?
  - **Partition of Data and Functions** If there is a discontinuity between devices, a particular service is not available for its entirety on some devices, is this apparent to the user?
  - **Procedures** Same functionality on different devices should be accomplished in the same way.
- **Task Continuity** All the different devices are aware of the actions done in the system by the user, irrespective with which device the action was performed. Task continuity usability points to watch are:
  - **Recovery of State of Data** When an activity is interrupted and is resumed on another device, the state of the data should be presented in such a way which confirms to the expectations of the user.
  - **Recovery of Activity Context** When a user interrupts his activity, he leaves at a specific point in the application - for example the cursor in a text editing program at the end of an incomplete sentence. When he resumes the activity after a long enough period, he can forget his last action when the activity context is not saved. For example, the cursor is after the resume of the activity at the beginning of the document and the user forgets he was finishing a sentence.

> This is also a problem for one-device applications, but a multi device system can aggravate the problem.

These usability criteria are going to be used by the testing moderator when he observes the test participant when he performs the test actions.

### 6.1.3 *Test Setting*

The test participants perform their primary test work in an enclosed room, which is an office or which resembles an office room. The test moderator is present wherever the test participant is.

Before the actual test starts, the test participant is asked to fill in a questionnaire, which contains questions about his or her demographic details and questions to determine the technological level of proficiency of the participant. This questionnaire can be found in Appendix A.

After the pretest questionnaire, the user is given some information about the system. Explaining it is about activity promotion and that there are multiple devices involved. It is also told that it is possible to continue functionality from one device to the other.

When the test starts the participant is explained to think aloud his thoughts. The test moderator will give an example how this should be done if the test participant would feel inclined to. During the test, interesting thoughts and thoughts at key points are written down by the test moderator.

The test moderator also has an observation schema to help him focus on the key points of the tests. This schema can be found in Section 6.1.4.

At the end of the test, after each test task is performed, an informal discussion will be held on the impression of the system on the test participant. A post-test questionnaire will not be done, because we should have all the information we want from the observations.

If something was missed during the test, the test moderator asks for this in the discussion at the end of the test.

Questions by the user will first be echoed back. If the user does not succeed in the task, he will be given hints - which will be written down in the notes.

Each test is recorded with a camera or sound recorder, making it possible to not only review the test moderator's notes but also the test itself.

The test scenario will first be performed with a pilot test, testing if the evaluation tasks are logical and to check that there are no obvious errors in the test or system.

The number of test participants for the evaluation will be first set to five persons. Virzi states that approximately eighty percent of the usability problems are identified with five subjects [36]. This will be

enough for prototype testing since eighty percent of the problems identified are enough to develop the system further for a more stable and fleshed out system. When this is done, additional usability tests are necessary.

If it appears that the number of new problems detected with each new test participant do not drop to an acceptable level of three errors, extra tests can be done until this is the case.

The questionnaires which are part of the test and which the user receive on one of the devices, contains questions about means of transport to and from work, how rested the participant is, sports the participant partakes in and if the participant went on holiday.

Their goal was to test the system itself through the answering of the questionnaire. The answers given by the participant were not of any importance, but it was to give the participant the feeling he was using an activity promotion system.

The questions and answers can be found in Appendix B.

The two feedback messages given in the evaluation are also not based on the real activity performance of the user. This is done, because each evaluation takes a relative short time - around thirty minutes - too short to give a useful message. Activity measuring is also not a part of this thesis and the evaluation, since this already works correctly.

What was done is that the first message given is negative, urging the user to take a walk for example. The second feedback message was positive, congratulating the user on his activity level. Both messages were given at a predefined time.

### 6.1.4 *Test Scenario*

In this subsection we will describe each test task in chronological order, which use case it tests, the observational points the test moderator must take into account and other related notes.

| **Task 1** | **User turns on smartphone and desktop** |
|---|---|
| Description | The user is asked to turn the smartphone and desktop on. |
| Use case | Use case 1: Device sign on, Table 1 |
| Observational points | • **Procedures** Does the user feel it works the same on both devices? |
| Notes | None |

Table 12: Task 1: User turns on smartphone and desktop

| **Task 2** | **Check Internet sites and receiving feedback** |
|---|---|
| Description | The user is asked to check some Internet sites he would like. During this task he will receive a feedback message urging him to take a walk. |
| Use case | Use case 3: User handover, Table 3<br>Use case 4: Feedback, Table 4 |
| Observational points | • **Attentiveness** Does the user note the feedback message.<br>• **Portability** How portable is the smartphone.<br>• **Response** How does the user respond to the feedback message. Will he take a walk or not? |
| Notes | If the user would leave the desktop computer, does the user handover happen in a correct way with regards to timing? |

Table 13: Task 2: Check Internet sites and receiving feedback

| **Task 3** | **Receiving a questionnaire and answering it.** |
|---|---|
| Description | Depending on if the user is taking a walk or is still behind the desktop computer, a questionnaire is send to the smartphone or desktop computer. The user is asked to fill in a questionnaire.<br>When after a couple of questions the user has not switched the realizers by himself, the test moderator will tell him that this is possible.<br>The user can switch between devices if he wants to. |
| Use case | Use case 5: Questionnaire, Table 5<br>Use case 6: Realizer switch in questionnaire, Table 6<br>Use case 7: User handover during questionnaire, Table 7 |
| Observational points | • **Attentiveness** Does the user note the questionnaire. Does the user note that it is possible to switch realizers.<br>• **Task Continuity** In case of a user handover, are the question and answers synchronized correctly? Does the user handover takes place in such a way the user would expect it.<br>• **Portability** How portable is the smartphone.<br>• **Partition of Data and Functions** When the user has experienced answering questions with his voice on the smartphone and he continuous on the desktop, does he realize it is now not possible to answer the questions with his voice?<br>• **Response** How does the user respond to the questionnaire. Will he switch between devices? |
| Notes | How does the user react to the different realizers and do they function in the way the user expects.<br>Does the speech recognition work correctly and does it work in the way expected by the user. |

Table 14: Task 3: Receiving a questionnaire and answering it

| **Task 4** | **Walking and receiving a feedback message** |
|---|---|
| Description | The user is asked to take a walk, during which he will receive a feedback message |
| Use case | Use case 4: Feedback, Table 5 |
| Observational points | • **Attentiveness** Does the user note the feedback message on the smart phone.<br>• **Portability** How portable is the smartphone. |
| Notes | None |

Table 15: Task 4: Walking and receiving a feedback message

| **Task 5** | **Receiving questionnaire reminder at the kiosk** |
|---|---|
| Description | The user is asked to go get a cup of coffee. Next to the coffee machine the kiosk display is placed.<br>When the user is near the coffee machine, he receives a questionnaire.<br>A reminder for the questionnaire is now visible at the kiosk display.<br>When the user previously has not performed the questionnaire at the desktop computer, he is asked to fill it on the desktop computer.<br>If he has not filled it in on the smartphone, he is asked to fill in the questionnaire on the smartphone.<br>It is left to the user to switch realizers. |
| Use case | Use case 3: User handover, Table 3<br>Use case 5: Questionnaire, Table 5<br>Use case 6: Realizer switch in questionnaire, Table 6<br>Use case 7: User handover during questionnaire, Table 7 |
| Observational points | • **Attentiveness** Does the user note the reminder on the kiosk.<br>• **Task Continuity** In case of a user handover, are the question and answers synchronized correctly? Does the user handover takes place in such a way the user would expect it.<br>• **Procedures** Does the user think that filling in the questionnaire is the same on both devices?<br>• **Partition of Data and Functions** When the user has experienced answering questions with his voice on the smartphone and he answers the question on the desktop, does he realize it is not possible to answer the questions with his voice?<br>• **Portability** How portable is the smartphone. |
| Notes | Does the speech recognition work correctly and does it work in the way expected by the user. |

Table 16: Task 5: Receiving questionnaire at the kiosk

| **Task 6** | **Closing and exiting the desktop computer and the smartphone.** |
|---|---|
| Description | The user is asked to close and exit the desktop computer and the smartphone. |
| Use case | Use case 2: Device sign off, Table 2 |
| Observational points | • **Procedures** Does the user feel it works the same on both devices? |
| Notes | None |

Table 17: Task 6: Closing and exiting the desktop computer and the smartphone

### General observation points

The following points should be observed for all test tasks:

- **Manageability** How easily can the user use the different GUI's of the different devices.
- **Learnability** How much help does the user need in performing the test tasks.
- **Knowledge Continuity** Are the interfaces on the different devices in such a way the user expects it. And does the user think that the same functionality can be done in comparable ways on the different devices?

## 6.2 RESULTS

The tests were done at the office complex of Roessingh Research and Development over a course of two days. Six participants took the test, of which four are female and two male. Their ages are between 24 and 54, five of them work in an office and the other is a student. They rate themselves as being proficiently with technology and three of the participants have a smartphone. The participants which have a smartphone say they (almost) always have it with them.

The test participants were told to surf the Internet, read some websites they like, check the mail and so on. They knew something would happen with regards to receiving activity feedback and questionnaires. Also they were told that they would get instructions to get some coffee at the coffee corner.

During the tests the system worked correctly, although there were in two cases problems with the Internet connection, resulting in an incorrect working of the Kiosk device. This shows that a robust Internet or connection handling is mandatory.

The use cases 1 and 6 where done by the test moderator, because this could not be done in a user friendly way.

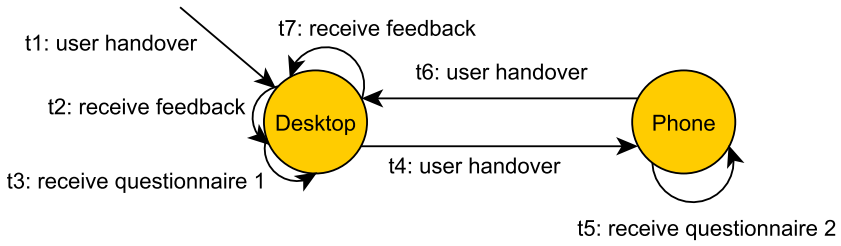


Figure 31: Activity flow of test participant 1

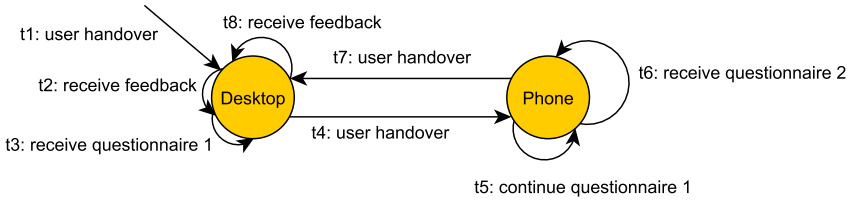


Figure 32: Activity flow of test participant 2

In Figure 31 to Figure 36, the activity flows of all the test participants can be seen. All the import system actions, such as user handover, questionnaire and feedback, are visible. The Kiosk device is not in the flows, because the user does not interact with it. Whenever a questionnaire is taking place, the Kiosk device shows the reminder.

We have learned the following points from the tests and the test participants:

- **User detection desktop** As can be seen in Figure 34, there are a lot of user handovers in the beginning. This is the result of the

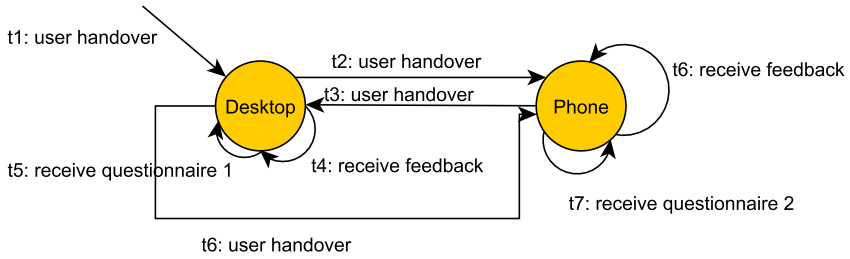


Figure 33: Activity flow of test participant 3

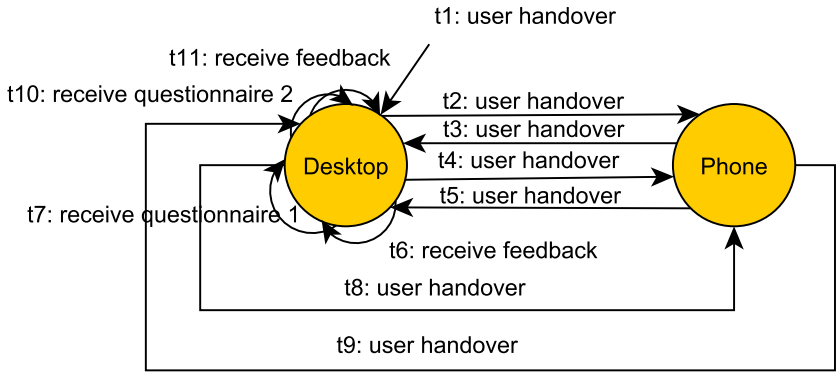


Figure 34: Activity flow of test participant 4

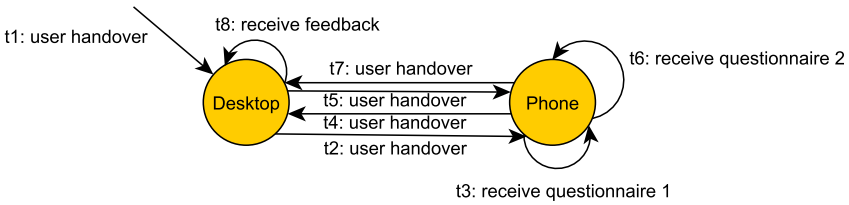


Figure 35: Activity flow of test participant 5

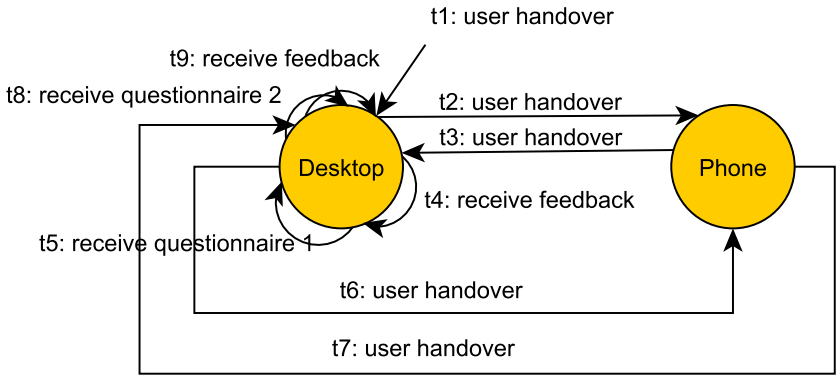


Figure 36: Activity flow of test participant 6

user reading a website and not using the mouse or keyboard for a while, selecting another page on the website, not doing anything for awhile, etcetera.

Now it is possible that the user can receive a message at the smartphone, although he is actually engaged with the desktop - resulting in a lower attentiveness to the smartphone and message.

An other way of detecting the user at the desktop is thus necessary, for example with bluetooth or a NFC chip. Although if we only use this solution it is possible that we would assign the user object to the desktop when in reality he is engaging with the smartphone[1].

- **History** Two of the participants would like to have the ability to view previous feedback messages and to edit or check answers given in questionnaires.

- **Portability smartphone** Three of the six participants needed to be reminded to take the smartphone with them when getting coffee and an other participant said she normally would not take her phone to the coffee machine. This means that detecting the user at the coffee machine with the use of the smartphone is not a good detection technique, but what kind of technique is there left?

  It is possible to use a clip with a chip which is small enough to be carried unobtrusively, but this should also contain an activity sensor. We do not know if it is technically feasible to make such a small activity sensor.

- **Attentiveness** The users saw the questionnaires and feedback messages both on the desktop and smartphone when they look at the device. During the test, one user placed the smartphone in his pocket. When it was in his pocket a questionnaire came on, but he did not notice it. This was when he was at the coffee corner and because of the kiosk display he knew that there was a questionnaire. A vibration of the smartphone could have helped, but this is also not always noticed.

- **Speech** The speech recognition worked adequately, although sometimes it was necessary that users repeated themselves. This should not happen too often or users would deem this input method unreliable and would not use it anymore.

- **Realizer switching** During a questionnaire four of the six users did not switch the realizers, even though they were told that

1 And it is possible to use the smartphone also for other things, besides the activity promotion application.

they could do this. The participants were mostly focused on the questions themselves. The user who switched between realizers the most knows the standard C3PO application and she was eager to explore the new options.

- **Questionnaire** Show that the questionnaire is ended, otherwise users can wait needlessly.
- **User preferences** Two of the six users liked to do the questionnaires on the smartphone and one user had a preference to do it on the desktop. The other three users did not mind on which device it took place.

# 7 CONCLUSIONS

## 7.1 DISCUSSION

The created system displays it is possible and feasible to create a multi device system for the Continuous Care & Coaching Platform (C3PO). A first prototype has been built which can be put into use with some adaptations, regarding robustness, security and support for multiple users.

In Section 1.2 we have listed the three goals we had and the challenges we expected to encounter. We will list them by name and discuss how they were handled and perceived by the test participants.

First we will discuss the challenges presented by Segerstahl [31].

- **Heterogeneity** During the tests it showed that some users have a preference for the desktop or smartphone device. When more devices are added, users should have or receive experience in using them - otherwise the system is less effective. This is a point which holds for each multi device or cross media system.
- **Interoperability** Each device has the same functional architecture, each application is build up out of modules which talk to each other via the local Hub. This makes it possible to easily create new kinds of applications, applications (or devices) which can inter-operate easily with each other.
- **Consistency** It was relative easy to create a consistent cross media platform. There are two reasons for this, the first being that new functionality was developed by our selves and secondly because we could use and extend the architecture developed in C3PO by Roessingh Research and Development[1]. The test users did not notice a difference in interaction logic between the different devices.

The usability challenges discussed by Oquist et al. [27] will be discussed next.

- **Portability** Some users expressed or showed during the tests that they do not take the smartphone with them for short walks or when they leave their workplace for a short time. This reduces the accuracy of the activity tracking. This can only be solved when they can carry something small with them and which they also do not have to remove from their pants when

1 http://www.rrd.nl/

they are sitting. They should not be given the opportunity to forget the device.

- **Attentiveness** The feedback and questionnaires are easily seen by the users on the different devices when they look at it. When the smartphone was in a pocket in a test case, the questionnaire was noticed with help of the kiosk device. Without the kiosk device it would have gone unnoticed until the user would have taken the smartphone out of his pocket or if a user handover would take place.

  Using vibration to get the user's attention is possible the only thing which can be done extra.

- **Manageability** The test users could use the GUI's easily with some explanation at the start of the test.

- **Learnability** During the tests there was not a problem with understanding how to interact with the devices after the users received an explanation in the beginning. During the questionnaires some users needed to be reminded that they can switch realizers, although even then most of them did not do this.

Now the usability challenges for cross media systems given by Denis et al. [10] will follow.

- **Knowledge Continuity**
  - **Visual appearance and continuity** The test users did not have problems with the interfaces, they found it logical how everything was presented to them.
  - **Partition of Data and Functions** During the test it was not clear to one user in the beginning he could not use speech input at the desktop. Speech input should be added to the desktop device and application, otherwise it should be made more clear that there is not any speech input at the desktop.
  - **Procedures** There were no problems with accomplishing the same functionality on the different devices.

- **Task Continuity**
  - **Recovery of State of Data** Questionnaires are synchronized between devices, thus a recovery of state of data takes place. This was also done in a logical way according to the test users actions.
  - **Recovery of Activity Context** This was not a point which came up during the tests, since the questionnaires and the test themselves were not long enough that a case would arise where a user would interrupt and forget an activity.

The three goals we set in Section 1.2 were all met. C3PO is extended in a cross media system, an unlimited number of devices can be run which can run instances of the smartphone, desktop or kiosk application.

The second goal was regarding the possibility of a user handover, which works for any number of devices.

The third goal was to have dialog capabilities in the specific applications, which is also implemented.

## 7.2 FUTURE WORK

In this section we will discuss what can be improved and what other possibilities there are.

### 7.2.1 *User detection*

This subsection will discuss ways on how to improve user detection.

- **Bluetooth / Near Field Communication (NFC)** Support for Bluetooth or NFC with regard to detecting users in the vicinity of a device probably gives better results than detecting input on the device. This is especially necessary for the kiosk application, because users will not interact directly with it.
- **Camera based user detection** It can also be interesting to develop user detection with a camera, whether it is mounted at a desktop screen or if it is in the smartphone. When it is detected that the user looks at the camera, the device which has the specific camera is probably best suited to have the user object.
- **Special sensors on user** When there are special sensors in the user's shoes, it is possible to detect if he is walking or sitting. When he is walking he probably can only give his attention to the smartphone. If he sits the desktop computer is probably the best pick.

With all these suggestions user studies should be done in order to determine which combination or weighing of the different detection possibilities give the best result. It should also be noted that it is possible the user feels that his privacy is invaded when the user detection is too obtrusive, for example in the case of camera's.

### 7.2.2 *Application and devices*

This subsection will discuss ways on how to improve the use of devices and the applications running on the devices.

- **Context awareness of realizer manager** When the smartphone has the user object, and normally the avatar or speech realizer is used, it can be interesting to detect if the user is in a loud environment. When this is the case, the user can probably not hear it correctly, other people can take offense and, but not unimportantly, the user may not want sound output for privacy concerns. Thus when a loud environment is detected, the realizer manager should switch to an unobtrusive realizer. Another possibility is to pass this information to the dialog manager, when the user is in a loud environment he is on the move and maybe does not have time to answer questions.

  Context awareness can and should be done with other factors besides sound or noise. When other factors are taken into account it can be made more effectively.

- **Security** Communicating with the server should be done in a secure way, make it impossible to eavesdrop.

- **Input and output realizers on different devices** Make it possible to run the input and output realizer on different devices, in effect creating a remote control.

- **Kiosk application** Create more functionality for the kiosk application, making it more interesting and useful for the user.

- **Smartphone application** Make it possible to run other applications besides the smartphone application, making the smartphone more useful for users. Although in some cases this might not be desirable, because the target group could possibly not understand running a full operating system.

### 7.2.3 *Server*

This subsection will discuss ways on how to improve the server.

- **Update newly registered devices** The server should keep a log of the current dialog stack, or all dialog - in order to send it to a newly registered device. It needs to be determined how large this backlog should be, several hours, days or weeks? Or should the devices let the server take care of the data storage, but this compromises the application philosophy of Roessingh Research and Development. This philosophy entails that a device should be as independent as possible.

- **Multiple users** Support for multiple users. The system supports at the moment one user, but it is designed in such a way that it is easily possible to support multiple users.

- **Internet connection** Do not assume there is always an Internet connection, make message queuing and receiving more robust. Also make the system as a whole more robust by taking server crashes into account. Also try to make devices useful in case of connection loss, this is done for the smart phone - since the smart phone has the user object when there is no connection - but what about the kiosk or desktop application?

- **User handover rules** Make it easier to add or edit user object handover rules in the server, preferably in such a way that a server restart is not necessary.

## 7.3 CLOSING

When creating a version 2 or a commercial version of the multi device C3PO system, there are some parts which needs to be done first.

Firstly, the system should be made more robust. As already said in Section 7.2, it should be taken into account that the server can go down or that the Internet connection is lost. When there is no Internet or when the server is down, it should be decided on which devices the application should start. The question then is, which applications have a use offline. These applications should also buffer the information they receive which they can send to the server when it is up again or when a connection is established.

Secondly, the security of the system should be improved. The connection with the server itself should be encrypted as well as access on the server to the details of the user.

The third part of improvement is multiple user support. At the moment the devices and server assume there is only one user, all the information they receive is grouped together. When information is sent and received, it should be associated with a user.

For testing purposes, the tests done in this thesis can be repeated to quickly get a feel how the users would react to the improved system. The place for this kind of test is in the development phase. When the system reaches towards completion it is important to do longer tests, which spans a day or multiple days.

Longer tests are necessary since the user uses the system a long time, months are not out of the question depending on the kind of therapy or other uses. This needs to be simulated and it should be found out how users react to the system when they use it for a long time.

The Thinking Aloud method is then also not a correct evaluation method, users should take notes in a diary or should receive in depth interviews.

All in all, the delivered prototype shows it is possible to create a multi device C3PO system, which works correctly and which can already be used for demonstration purposes.

# BIBLIOGRAPHY


[1] Apache ActiveMQ TM – Index., 2012. URL http://activemq.apache.org/.

[2] OAuth Community Site., 2012. URL http://oauth.net/.

[3] SMARCOS: Main Page., 2012. URL www.smarcos-project.eu.

[4] SMARCOS: Main Page, 2012. URL http://www.smarcos-project.eu/smarcosSummary.html.

[5] Carol .M. Barnum. *Usability Testing and Research*. Longman, New York, New York, USA, 2002. ISBN 0-205-31519-4.

[6] Timothy Bickmore and Toni Giorgino. Health dialog systems for patients and consumers. *Journal of biomedical informatics*, 39(5): 556–71, October 2006. ISSN 1532-0480. doi: 10.1016/j.jbi.2005.12.004. URL http://www.ncbi.nlm.nih.gov/pubmed/16464643.

[7] J Boumans. Cross-media E-Content Report 8. *ACTeN-Anticipating Content Technology Needs*, (August):1–21, 2004. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Cross+Media:+E-content+report+8#0.

[8] Sunny Consolvo and DW McDonald. Theory-driven design strategies for technologies that support behavior change in everyday life. *Proceedings of the 27th*, pages 405–414, 2009. URL http://dl.acm.org/citation.cfm?id=1518766.

[9] Fiorella de Rosis, Nicole Novielli, Valeria Carofiglio, Addolorata Cavalluzzi, and Berardina De Carolis. User modeling and adaptation in health promotion dialogs with an animated character. *Journal of biomedical informatics*, 39(5):514–31, October 2006. ISSN 1532-0480. doi: 10.1016/j.jbi.2006.01.001. URL http://www.ncbi.nlm.nih.gov/pubmed/16524784.

[10] Charles Denis and Laurent Karsenty. Inter-Usability of Multi-Device Systems - A Conceptual Framework. In Ahmed Seffah and Homa Javahery, editors, *Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces*, chapter 17, pages 373–385. Wiley, 1 edition, 2004. ISBN 978-0470854440.

[11] Carol Ewing Garber, Bryan Blissmer, Michael R Deschenes, Barry a Franklin, Michael J Lamonte, I-Min Lee, David C Nieman, and David P Swain. American College of Sports Medicine position stand. Quantity and quality of exercise for developing

and maintaining cardiorespiratory, musculoskeletal, and neuromotor fitness in apparently healthy adults: guidance for prescribing exercise. *Medicine and science in sports and exercise*, 43(7):1334–59, July 2011. ISSN 1530-0315. doi: 10.1249/MSS.0b013e318213fefb. URL http://www.ncbi.nlm.nih.gov/pubmed/21694556.

[12] Jukka Honkola, Hannu Laine, Ronald Brown, and Ian Oliver. Cross-Domain Interoperability : a Case Study. In Sergey Balandin, Dmitri Moltchanov, and Yevgeni Koucheryavy, editors, *Smart Spaces and Next Generation Wired/Wireless Networking*, pages 22–31. Springer Berlin / Heidelberg, 2009. doi: 10.1007/978-3-642-04190-7_3. URL http://dx.doi.org/10.1007/978-3-642-04190-7_3.

[13] Robert Hurling, Michael Catt, Marco De Boni, Bruce William Fairley, Tina Hurst, Peter Murray, Alannah Richardson, and Jaspreet Singh Sodhi. Using internet and mobile phone technology to deliver an automated physical activity program: randomized controlled trial. *Journal of medical Internet research*, 9 (2):e7, January 2007. ISSN 1438-8871. doi: 10.2196/jmir.9.2.e7. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1874722&tool=pmcentrez&rendertype=abstract.

[14] Daniel Jurafsky and James H Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall Series in Artificial Intelligence. Pearson Education, 2nd edition, 2008.

[15] J. Kiljander, M. Etelapera, J. Takalo-Mattila, and J. Soininen. Opening information of low capacity embedded systems for Smart Spaces. In *Intelligent Solutions in Embedded Systems (WISES), 2010 8th Workshop on Intelligent Solutions in Embedded Systems*, pages 23 – 28, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5548438&tag=1.

[16] Randy Klaassen, Jordi Hendrix, Dennis Reidsma, and Rieks Op den Akker. Elckerlyc goes Mobile Enabling Technology for ECAs in Mobile Applications. 2012.

[17] Willemieke Kroeze, Andrea Werkman, and Johannes Brug. A systematic review of randomized trials on the effectiveness of computer-tailored education on physical activity and dietary behaviors. *Annals of behavioral medicine : a publication of the Society of Behavioral Medicine*, 31(3):205–23, June 2006. ISSN 0883-6612. doi: 10.1207/s15324796abm3103_2. URL http://www.ncbi.nlm.nih.gov/pubmed/16700634.

[18] James A Larson. Standards VoiceXML and the W3C Speech Interface Framework. *IEEE Multimedia*, 10(4):91–93, 2003.

[19] Petri Liuha, Juha-pekka Soininen, and Raul Otaolea. SOFIA : Opening embedded information for smart applications. In *Intelligent Solutions in Embedded Systems (WISES), 2010 8th Workshop on Intelligent Solutions in Embedded Systems*, Heraklion, 2010.

[20] Leszek A. Maciaszek. *Requirements Analysis and System Design*. Pearson Education Limited, Essex, first edition, 2001. ISBN 0201709449.

[21] MF McTear. Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit. *Fifth International Conference on Spoken Language ...*, 1998. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Modelling+spoken+dialogues+with+state+transition+diagrams+:+experiences+with+the+CSLU+toolkit#0.

[22] Jeffrey P Migneault, Ramesh Farzanfar, Julie a Wright, and Robert H Friedman. How to write health dialog for a talking computer. *Journal of biomedical informatics*, 39(5):468–81, October 2006. ISSN 1532-0480. doi: 10.1016/j.jbi.2006.02.009. URL http://www.ncbi.nlm.nih.gov/pubmed/16564749.

[23] Leonie Michelle Neville, Blythe O'Hara, and Andrew Milat. Computer-tailored physical activity behavior change interventions targeting adults: a systematic review. *The international journal of behavioral nutrition and physical activity*, 6:30, January 2009. ISSN 1479-5868. doi: 10.1186/1479-5868-6-30. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2700068&tool=pmcentrez&rendertype=abstract.

[24] D.A. Norman. *The Invisible Computer. Why good products can fail, the personal computer is so complex and information appliances are the solution.* MIT Press, Cambridge, 1998.

[25] Harm Op den Akker, Monique Tabak, Mihai Marin-Perianu, Rianne Huis in't Veld, Valerie M Jones, Dennis Hofs, Thijs M Tonis, Boris W Van Schooten, Miriam M.R. Vollenbroek-Hutten, and Hermie J Hermens. Development and Evaluation of a Sensor-Based System for Remote Monitoring and Treatment of Chronic Diseases : the Continuous Care & Coaching Platform. In *Proceedings of EHST 2012: the 6th International Symposium on eHealth Services and Technologies*, Geneva, Switzerland, 2012.

[26] Rieks Op den Akker, Randy Klaassen, T Lavrysen, G Geleijnse, A Van Halteren, H Schwietert, and Marlies Van der Hout. A Personal Context-Aware Multi-Device Coaching Service that Supports a Healthy Lifestyle. In *Proceedings of HCI 2011*, pages 1–6, Newcastle Upon Tyne, 2011. URL http://eprints.eemcs.utwente.nl/20810/01/HCI2011_Smarcos_WP4_SUBMITTED.pdf.

[27] Gustav Oquist, Mikael Goldstein, and Didier Chincholle. Assessing Usability across Multiple User Interfaces. In Ahmed Seffah and H. Javehery, editors, *Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces*, chapter 15, pages 325–349. John Wiley & Sons, Ltd, 2005. ISBN 9780470854440. doi: 10.1002/0470091703.ch15. URL http://dx.doi.org/10.1002/0470091703.ch15.

[28] Terry Quatrani. *Visual modeling with Rational Rose 2000 and UML*. Addison Wesley, 2000. ISBN 0201729326. URL http://books.google.com/books?hl=en&lr=&id=SFGngeydCQgC&oi=fnd&pg=PR13&dq=Visual+modeling+with+Rational+Rose+2000+and+UML&ots=CPx9WP-HHw&sig=2OjG_Y8jnWY1ocs0NcoYNR0vKNk.

[29] Jeffrey Rubin and Dana Chisnell. *Handbook of usability testing: how to plan, design, and conduct effective tests*. Wiley Publishing, Indianapolis, second edi edition, 2008. ISBN 978-0-470-18548-3.

[30] Ichiro Satoh. MobileSpaces: A framework for building adaptive distributed applications using a hierarchical mobile agent system. *Distributed Computing Systems, 2000. Proceedings.*, 2000. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=840918.

[31] Katarina Segerstahl. Utilization of pervasive IT compromised? In *Proceedings of the 7th International Conference on Mobile and Ubiquitous Multimedia - MUM '08*, page 168, New York, New York, USA, 2008. ACM Press. ISBN 9781605581927. doi: 10.1145/1543137.1543171. URL http://portal.acm.org/citation.cfm?doid=1543137.1543171.

[32] Katarina Segerstahl. Crossmedia Systems Constructed around Human Activities: A Field Study and Implications for Design. 2009. URL http://www.springerlink.com/index/31V5W310H2U44168.pdf.

[33] Bruce Snyder, Dejan Bosanac, and Rob Davies. Introduction to Apache ActiveMQ. In *Active MQ in Action*, chapter 1, pages 6–16. Manning Publications Co, 1 edition, 2012. ISBN 1933988940.

[34] Tessella. Annual report 2006-2007, 2007. URL http://www.tessella.com/wp-content/uploads/2008/05/annualreport2006_2007.pdf.

[35] Maaike van den Haak, Menno De Jong, and Peter Jan Schellens. Retrospective vs. concurrent think-aloud protocols: Testing the usability of an online library catalogue. *Behaviour & Information Technology*, 22(5):339–351, September 2003. ISSN 0144-929X. doi: 10.1080/0044929031000. URL http://www.tandfonline.com/doi/abs/10.1080/0044929031000.

[36] RA Virzi. Refining the test phase of usability evaluation: How many subjects is enough? *Human Factors: The Journal of the Human Factors and ...*, 34(4):457–468, 1992. URL http://hfs.sagepub.com/content/34/4/457.short.

[37] Joseph Weizenbaum. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, January 1966. ISSN 00010782. doi: 10.1145/365153.365168. URL http://portal.acm.org/citation.cfm?doid=365153.365168.

[38] WA Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, pages 591–606, 1970. URL http://dl.acm.org/citation.cfm?id=362773.

[39] World Health Organization. Reducing Risks, Promoting Healthy Life, 2002. URL http://www.who.int/whr/2002/en/.

[40] World Health Organization. WHO | Physical Inactivity: A Global Public Health Problem, 2012. URL www.who.int/dietphysicalactivity/factsheet_inactivity/en/.

Part I

# APPENDIX

# A

# QUESTIONNAIRE

## A.1 VRAGENLIJST

| | |
|---|---|
| Leeftijd | ..... |
| Beroep | ..... |
| Geslacht | ..... |

| | |
|---|---|
| Ik heb een computer thuis: | ja/nee |
| Ik gebruik een computer op het werk: | ja/nee |
| Ik heb een smartphone: | ja/nee |
| Als ja; die heb ik (bijna) altijd bij mij: | ja/nee |
| Ik kan apps op een smartphone starten: | ja/nee |
| Ik vind dat ik vaardig met technologie ben: : | ja/nee |

# B

# TEST QUESTIONNAIRE

This Appendix contains the questionnaires which were part of the evaluation and which the user received on a device. Their goal was to test the system itself through the answering of the questionnaire.

Numbers behind an answer indicate the number of the following question.

## B.1 QUESTIONNAIRE 1

1. Hoe kwam u vandaag naar het werk?
   a) Fiets
   b) Auto
   c) Lopen
   d) Bus
2. Voelt u zich uitgerust?
   a) Ja
   b) Redelijk
   c) Matig
   d) Nee
3. Heeft u gisteren gesport?
   a) Ja (4)
   b) Nee (5)
4. Wat voor soort sport was het?
   a) Fitness
   b) Balsport
   c) Hardlopen
   d) Vechtsport
5. Hoe vaak sport u per week?
   a) 0 keer
   b) 1 keer
   c) 2 keer
   d) 3+ keer
6. Hoe vaak per dag gaat u weg achter uw computer?

a) 1 tot 3 keer
b) 3 tot 6 keer
c) 6 tot 9 keer
d) 9 keer of meer

## B.2 QUESTIONNAIRE 2

1. Voelt u zich uitgerust?
   a) Ja (3)
   b) Redelijk (3)
   c) Matig (2)
   d) Nee (2)
2. Waardoor komt dit?
   a) Te weinig slaap
   b) Te veel gedaan
   c) Te warm
3. Hoe gaat u straks van het werk?
   a) Fiets
   b) Auto
   c) Lopen
   d) Bus
4. Heeft u al vakantie gehad?
   a) Ja (5)
   b) Nee (Einde)
5. Wat voor soort vakantie was het?
   a) Relax vakantie
   b) Actieve vakantie
   c) Reizen naar verschillende plekken