Capturing the attention of a mobile user by sending information to the right output device

Erik Lövbom

August 18, 2008

Master’s Thesis in Computing Science, 30 ECTS credits
Supervisor at Department of Computing Science: Thomas Pederson
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

During real world activities there is always the risk that interaction with a computer inhibits a person’s ongoing activities. It is also difficult for a computer system to attract the user’s attention during such activities. This master thesis presents a theoretical design and a prototype for a system that addresses these two problems.

In the easy-activities-of-daily-living (easyADL) project the idea of a situative space model was developed to address the fact that objects close to the user are of greater importance than objects far away from the user. This master’s thesis (as part of the easyADL project) presents a simulation environment that makes use of the situative space model for filtering and ranking objects in the vicinity of a mobile user.

Here we assume a mobile user wearing a light-weight computer. A theoretical design is presented which involves a software component that selects the best modality, the best input, and the best output devices for communication between a given software application and the user, based on the user’s environment and ongoing activities.

A prototype was developed to demonstrate the theoretical design of this system. To run the prototype some other software programs also had to be developed, and those are also integrated in the simulation environment.

Two different versions of the prototype were evaluated by six subjects. One prototype automatically selected the most appropriate device -- a device visible to the subject at the time. The other prototype broadcasted the information to all available output devices.

The subjects in this pilot study preferred messages which were broadcasted rather than those presented on a device in the user’s observable space. Voice messages were preferred to text messages. Two of the subjects suggested that an audio signal could be used to catch the user’s attention in combination with a text message presented on a display near the subject.

Table of contents

1. Introduction 1
2. Goals 3
   Requirements 3
   Problem description 4
3. Method 5
4. Related work 7
   4.1 How to get the user's attention 7
   4.2 Handling a dynamic number of resources 8
   4.3 How much information can the target group handle? 9
5. Design 11
   5.1 Scenario 11
      5.1.1 Cooking rice 11
   5.2 A situative space model 13
   5.3 Modalities 14
   5.4 Design discussion 15
      5.4.1 Display a message 15
      5.4.2 When shall the system stop displaying a message? 15
      5.4.3 Stop displaying a message 15
      5.4.4 Has the user read/heard the message? 15
   5.5 Design factors 16
      Contextual parameters to be considered by the Egocentric Interaction Manager 17
   5.6 Explicit input and output devices 18
   5.7 Egocentric Interaction Manager communication 19
   5.8 Choosing the best display 20
   5.9 Egocentric interaction manager functions 21
      5.9.1 Function parameters 21
      5.9.2 Functions 22
   5.10 Algorithms 23
      5.10.1 Select output device 23
      5.10.2 Select a display 23
      5.10.3 Send message to a display 23
      5.10.4 Select an audio device 24
      5.10.5 Send message to an audio device 25
      5.10.6 Remove a message 25
6. Implementation 27
   6.1. Deciding development environment 27
   6.2. The Egocentric Interaction Manager application 27
      6.2.1 The Egocentric Interaction Manager Class diagram 27
      6.2.2 EIM 29
      6.2.3 InputDeviceScopeListener 29
      6.2.4 OutputDeviceScopeListener 29
      6.2.5 SituativeSpaceMonitorListener 29
      6.2.6 CommandListener 29
      6.2.7 DisplayArray 35
      6.2.8 LoudspeakerArray 35
      6.2.9 KeyboardArray 35
      6.2.10 MicrophoneArray 35

6.3 Test applications 36
6.3.1 High level software application 36
6.3.2 Situative space monitor 37
6.3.3 Client applications 39
6.4 The system in action 40
7. Evaluation 43
7.1 Hypotheses 43
7.2 Experimental setup 43
7.3 Analysis 47
8. Conclusion 49
9. Future work 51
10. Acknowledgements 53
11. References 55
Appendix A – Study with the six persons that testran the simulation environment (in Swedish) 57

1. Introduction

The average age of citizens is increasing. With this, age-related healthcare is becoming a significant problem both from a humane and from an economic perspective. Many individuals wish to remain in their homes although they suffer from medical problems such as dementia [1].

EasyADL is a research project investigating new forms of computer technology for facilitating everyday life of individuals suffering from minimal to moderate dementia. The main goal of the project is to develop a prototype for a “cognitive prosthesis” which helps people suffering from cognitive disorders to perform important Activities of Daily Living (ADL) in their homes without professional support [2].

This master thesis project is connected to the research project easyADL. In easyADL, the cognitive prosthesis is implemented as a wearable computer system for user assistance. The system will run on a portable computer attached on the user’s body. The software on the portable computer does not need any user interface.

The task in this master’s thesis project was to design and implement a module in the cognitive prosthesis system, called the Egocentric Interaction Manager, see Figure 1, below. Details about the system will be presented in Chapter 5.

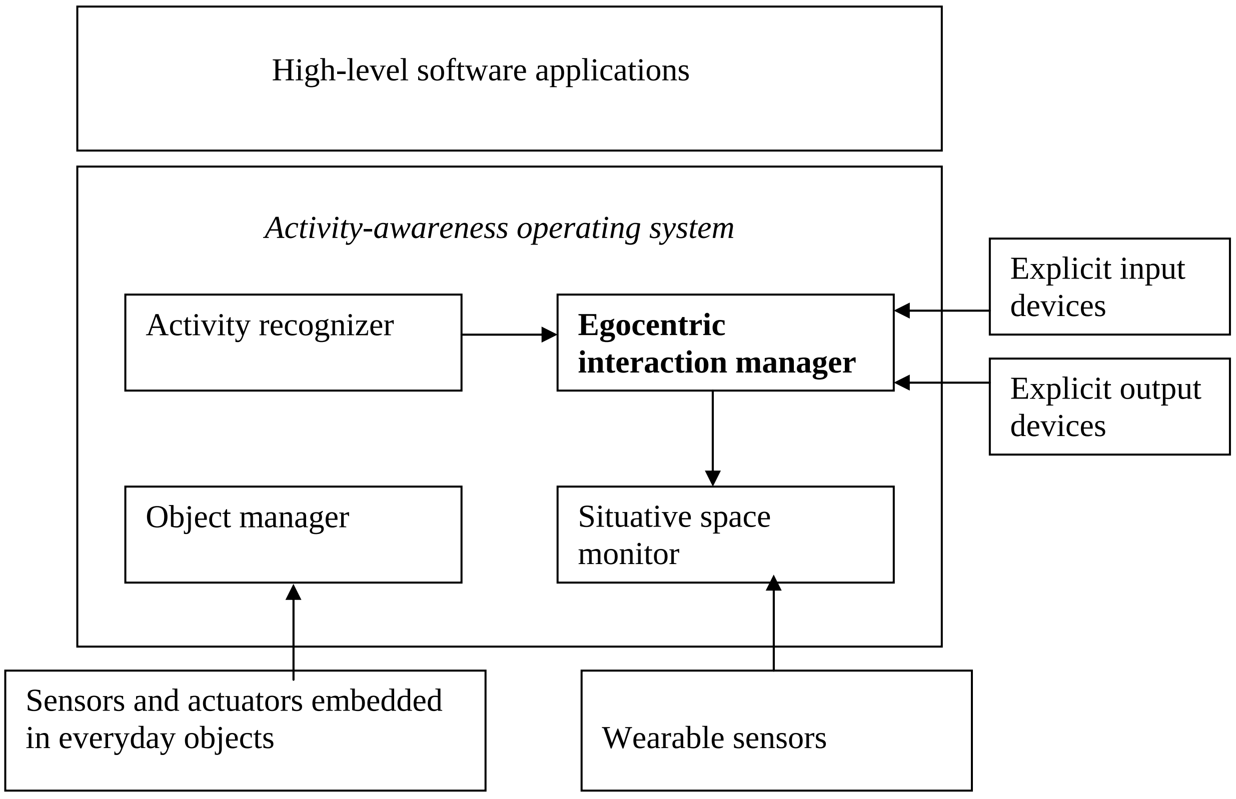

High-level software applications
Activity-awareness operating system
Activity recognizer
Egocentric interaction manager
Explicit input devices
Explicit output devices
Object manager
Situative space monitor
Sensors and actuators embedded in everyday objects
Wearable sensors

Figure 1: The activity-awareness operating system adapted from [3].

2. Goals

In the easyADL project the idea of a situative space model was developed to illustrate the fact that objects close to the user are of greater importance than objects far away from the user.

Goals for this master’s thesis:

1. To design a simulation environment that uses the situative space model for filtering and ranking objects in the vicinity of a mobile user.

2. To make a conceptual design of a software component that automatically selects the best modality and best input and output device for the communication between a given software application (running on a wearable computer) and the user, based on the user’s physical context.

3. To make a prototype software that will exhibit parts of the design.
   In order to run the prototype some other software modules in the system also have to be designed (see Figure 1):

4. To design a high-level software application which decides when and which information should be sent to the user.

5. To design a situative space monitor which stores information about the objects present in the user’s environment.

6. To design client applications with input and output devices such as keyboards, microphones, loudspeakers and displays

7. To simulate the user’s environment, an application with which one can move the user around in an apartment, is to be implemented.

Requirements

The software that was to be designed in this master’s thesis was required to be able to communicate with other software applications within the easyADL project (see Figure 1).

Problem description

There will be a dynamic number of input- and output devices connected to the system. Input devices refer to keyboards and microphones, output devices refer to loudspeakers and displays.

The software application to be designed needs to know which devices that are connected to the system (i.e. some sort of device discovery). It also needs to know which of these devices are close to the user. If the user is in the kitchen, it would be best to display information on a device located in the kitchen.

One issue concerns interruptability. For example, if the system is displaying a message to the user and another message needs to be displayed, can the system replace the current message with the next one? Does the system know if the user has read the current message? How shall the system handle the problem that there is no output device available when a message needs to be displayed?

Another issue is human attention. How can the system get the user’s attention? The system will run unnoticed in the user’s environment. Only when information needs to be displayed will the user’s attention be required.

What modality is best to enable communication between the user and the system in a given situation? The last question is related to the field of context awareness.

3. Method

In order to reach the goal of developing the interaction manager, as described in the goal description earlier, the following steps were taken:

- I looked for related work in ubiquitous computing in areas such as context awareness, human attention and dynamic resource management.
- A conceptual design of the software application was made.
- Development environment (for implementing the design) was chosen.
- The prototype and some other applications needed to run the system were implemented.
- The prototype was partially evaluated in the experiment.

4. Related work

There are several related works that look into ways of getting the user's attention, and how to handle a dynamic number of resources in a system.

4.1 How to get the user's attention

Traditionally, computer interfaces have been confined to conventional displays and focused activities. But as displays become embedded throughout our environment and daily lives, increasing numbers of them must operate on the periphery of our attention. Peripheral displays can allow a person to be aware of information while he or she is attending to some other primary task or activity [4].

How shall a software application get the user's attention? In the article by Matthews, T. et al [4] different notification (message) levels are defined. Higher notification levels correspond to more critical data and are displayed in a way that grabs a user's attention.

Lower notification levels correspond to non-critical data and are typically displayed in a way that does not attract attention, but allows a user to monitor a display occasionally or peripherally.

Displays attempting to attract **focal attention** for important information typically utilize abrupt transitions**.** Studies have shown that significant changes in the interface will draw a user's attention. As an example, rapidly changing sounds convey a sense of urgency more than smoother sounds. This was concluded in a survey carried out in a simulated softdrink factory environment where they used sound to augment events in the production [5].

Displays attempting to capitalize on **divided attention** employ a number of more subtle techniques, such as updating small pieces of the display abruptly or including slow movements.

For displays utilizing **inattention** it has been found that animations like fading, rolling, and tickering are not distracting to low-attention primary tasks. This suggests that repetitive and very gradual animations are appropriate for change blind transitions in this context.

According to Matthews, different types of animations are a key mechanism for supporting transitions in applications that do not want to distract users. Displays explicitly minimizing motion are more likely to be change-blind, and the display update goes unnoticed. However, no alternative to animation is presented in the article. Another survey focuses on usage of audio for both getting the user's attention and to deliver the message [5].

The concept of higher and lower notification levels was interesting. A similar thing in the egocentric interaction manager is to give messages different priorities (low, medium and high).

Another important concept is the usage of movements in displays to attract attention. Messages shown on a display without motion were more likely not to be noticed by the user. In the second article [5] usage of auditory cues was a good way to carry out a message because users need not to focus to hear a sound.

In the egocentric interaction manager voice is one modality. The good thing with voice messages is that the user need not focus on a display to get the information.

4.2 Handling a dynamic number of resources

How shall a software application know what devices are available in the user's vicinity? How shall it make use of a device services?

In **C-ANIS – A Contextual, Automatic and Dynamic Service-Oriented Integration Framework** [6], it is said that Ubiquitous computing environments are highly dynamic by nature. Services provided by different devices can appear and disappear, e.g., devices join and leave these environments. C-ANIS combines various services creating new services. An example is when a webcam service is integrated with a storage service; the new service will be a webcam and storage service.

C-ANIS proposes two different approaches to service integration: *automatic* integration and *on-demand* integration. Automatic integration automatically extends the functionality of an existing service *S* by integrating it with compatible services in the environment, but leaving the interface of *S* unchanged. This way, the extension in functionality of *S* can be kept transparent to applications or users employing this service. On-demand integration builds a new service on request from a list of given services in the environment, creating new interfaces. These new interfaces are employed at least by the users or applications which have requested their creation.

The research project **A software architecture for virtual device composition and its applications** has a similar approach [7]. They propose a software architecture enabling a user-centric virtual device that is a composition of the partial functions of surrounding devices.

Increasingly, new information appliances and mobile personal devices are being equipped with many primitive functions, such as network connectivity, small display, advanced user interface, etc. However, constraints like small screen and tiny keypad impose usage limitations for a user. In this project, the focus is on the possibility and flexibility of individual functions' unification and separation to create a new user friendly environment with personal and public devices.

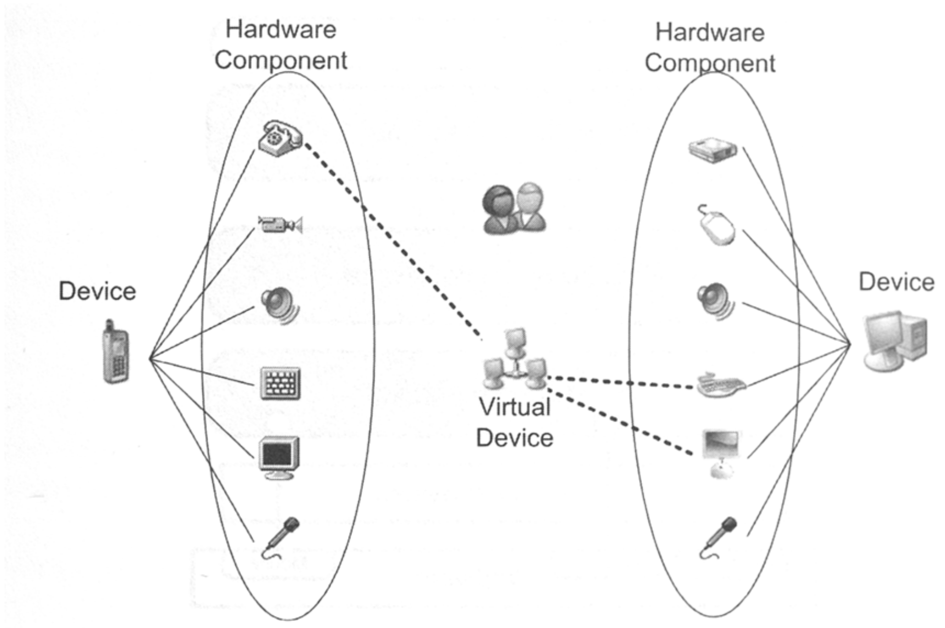


Figure 2 Concept of Virtual Device Composition. A virtual device is a logical device composed of a collection of multiple device functions. Each function of network connected devices can participate in a virtual device [7].

Devices in the user’s vicinity are discovered by a software manager on a user’s device (mobile phone / laptop). Discovered devices can be combined by the user to create a virtual device providing rich input or output capabilities. An example is when combining the phone call function of a mobile phone with the keyboard and display of a desktop PC, thus creating a virtual mobile phone with bigger display service and more convenient keyboard service than what is offered by the original mobile phone.

These projects have in common with EIM the fact that they view devices/ services as something that is highly dynamic -- they enter and leave the user’s environment. What differs is that EIM has a selection algorithm and to a further extent can act on behalf of itself.

4.3 How much information can the target group handle?

When displaying information the target group needs to be considered. Because the target group here is people with mild dementia it is important to keep information as short as possible, with a limited number of graphical icons [8].

From a previous work by Sjölund and Zingmark, the user’s interaction with the help system should be minimal, one thing at a time and not too much information [9].

5. Design

The software component that will be designed in this master's thesis is called **egocentric interaction manager** (EIM). The term 'egocentric' is used to indicate that it is the body and mind of a specific human actor that serves as the centre of reference to all his/ her interaction with everyday objects [10]. In the following text, the term 'system' refers to the activity-aware operating system being described on page 16 (Figure 9).

5.1 Scenario

How shall the system function ideally? In order to give the reader an introduction to how the system ideally can assist and help a user, a scenario in which the user is cooking rice is presented here.

5.1.1 Cooking rice

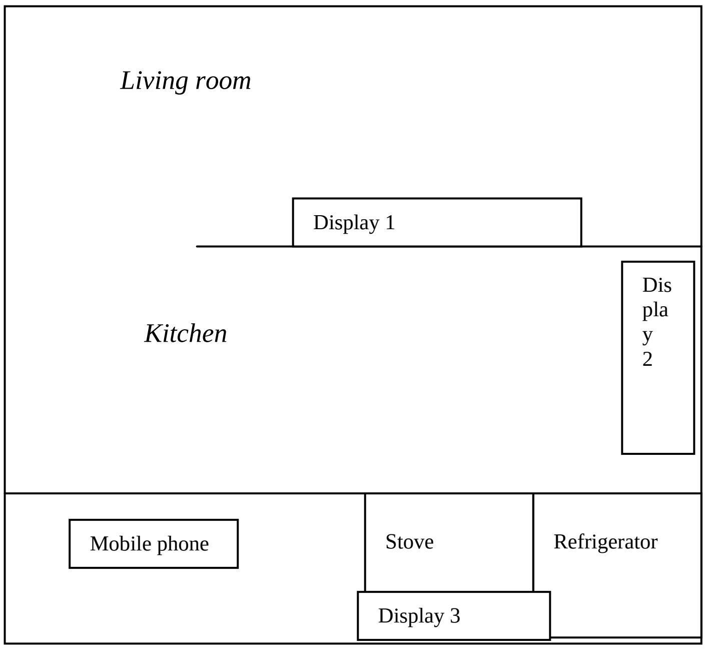


Figure3: A user's apartment.

User:	**Time:**	**The system:**
The user John picks up a rice cooker and puts it on the stove.		Notices that the user picks up and then releases a rice cooker.
John picks up a rice package.		Notices that a rice package is being picked by the user.
John puts rice in the rice cooker and releases the rice package.		The user is no longer holding the rice package.
John turns on the stove. Half a minute passes and John has still not put water into the rice cooker.		The stove is turned on. The user seems to be cooking rice but needs to be reminded to put water into the rice cooker. The user is in front of display 3 and this message will be shown on that display: "*John: You need to put water into the rice cooker*".
John notices the message on the display. He puts water into the cooker.	↓	The water tap is turned on and the turned off. The user picks up and then releases a can. As the system estimates that water is put into the cooker, the message on the display is removed.

Figure 4: Timeline for scenario cooking rice.

5.2 A situative space model

The thesis project will discuss objects and devices that are within immediate proximity of a human actor; therefore the concept of a situative space model needs to be outlined [10].

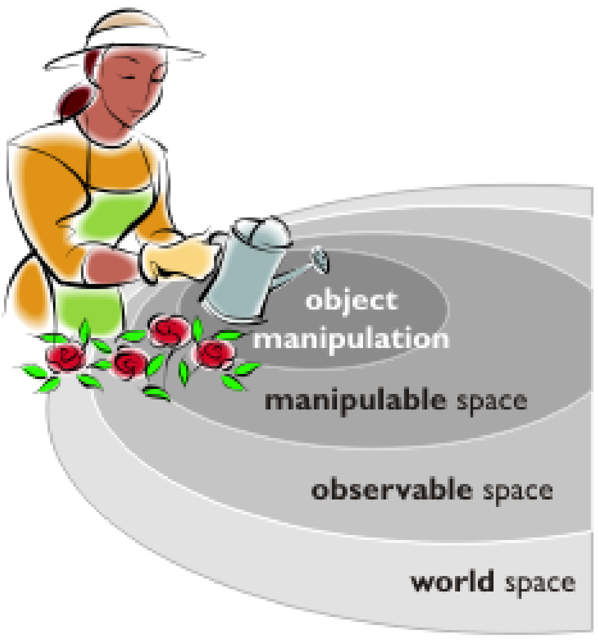


Figure 5: A situative space model[1].

A situative space model is developed on the basis of what a specific user can see/not see or touch/not touch at any given moment in time (Fig. 5).

Users situate themselves closer to the objects relevant for their current activity. Such explicit proximity gives an indication of the user's intent (the needs and wants of the user to satisfy some goal). By capturing the changes to the user's observable space and manipulable space, we are indirectly capturing the user's intentions [10].

The world space
This contains the set of all objects known to the system.

The observable space
The observable space is the set of objects within a cone in front of the user's eyes, and this cone follows the user's head movements as shown in Fig. 6(a). The height of the cone is limited by the walls in an indoor environment, and visual occlusion further affects the number of objects within this space. The software module to be designed in this master's thesis will pay attention to what objects are within the user's *observable space* and what objects are within the *manipulable space*.

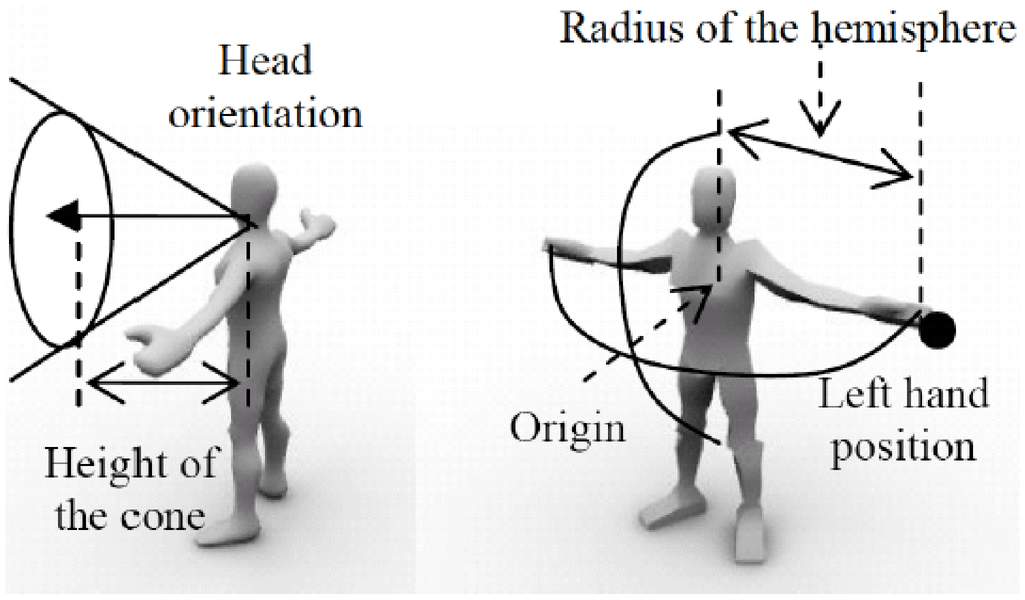


Figure 6(a) Observable space [10]. *(b) Manipulable space [10].*

The manipulable space
This is the set of objects within a hemisphere in front of the user’s chest as shown in Fig. 6(b). Such a shape is motivated by the fact that humans have two hands and the assumption that they manipulate objects within reach of their hands. The hemisphere follows the user’s chest movements, and its radius is equal to the maximum distance between the chest and a hand.
To put it simply, the objects that a user can touch are all within the manipulable space.

Object manipulation
In the easyADL project at the moment, object manipulation is when an object is touched and released by a user [10].

5.3 Modalities

This master’s thesis deals with several questions; how to choose modality for a message and how to find the best available output device for displaying a message. Here is a short presentation of the modalities used in this master’s thesis:

Voice
This type of message is a piece of information that can be played on an audio player device.

Text
Text information that can be shown on a display.

Image
A picture of an icon that can be shown on a display.

5.4 Design discussion

5.4.1 Display a message

A message is sent from a high level application to the EIM application (see Fig. 9). It is the EIM application's responsibility to decide modality and find the best output device on which to display the message.

A software application that wants to send a message through the use of EIM specifies the message modality (voice / text / image). If the modality is left unspecified, then EIM will decide what modality is best to use.

5.4.2 When shall the system stop displaying a message?

A *simple approach* is to let the message (either text or image) be visible until the next message is shown.

Confirmation button: The system will wait for the user to press a confirmation button (OK) and then remove the text or image.

A *time-based approach*: After a given amount of time has passed the message will be removed.

The *observable space approach*: When the user is in front of a display the message will be shown on that display. If the user moves so that another display is in front of him, the message will be shown on that display instead.

The *problem-fixed approach*: The situation causing the message is no longer there. If the message tells the user to turn off the stove and the stove is turned off, then there is no need to continue displaying the message.

5.4.3 Stop displaying a message

An obligatory parameter when calling the function for deactivating a message is *messageID*. With the messageID, the EIM application will inform each device that shows the actual message to stop showing it.

5.4.4 Has the user read/heard the message?

It can be difficult for the system to know if the user has read/heard the information being sent out. Here is a distinction between implicit input and explicit input.

Implicit input: The user is directed to turn off the stove, and the stove has been turned off. The system knows from the state of the stove, i.e., turned off, that the user has turned off the stove.

Explicit input: An example: the system wants to know if the user has brushed his teeth. A required feedback can be that the user replies in a microphone "Yes, I have brushed my teeth".

5.5 Design factors

Deciding which display is the most appropriate to use
Below is a picture with three different displays. Which one of them will best catch the user's attention? Which one would be most suitable for a specific activity, like reading a longer text?

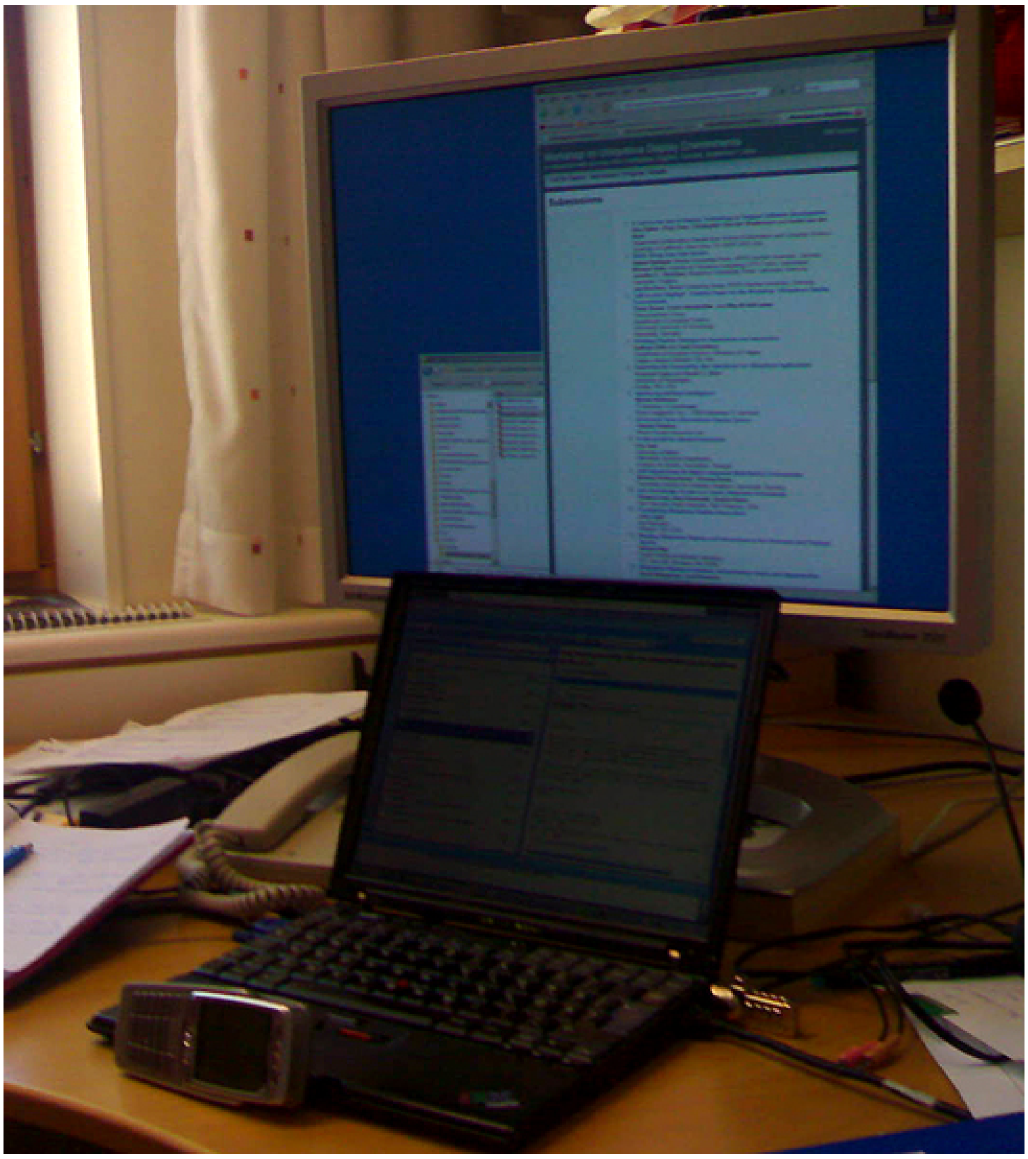


Figure 7: Three displays from which to choose: wide screen, laptop screen or mobile telephone screen.

Contextual parameters to be considered by the Egocentric Interaction Manager

User preferences
If the user prefers textual messages before voice messages or vice versa, it is an important factor in the process of selecting the most appropriate output device.

Field of vision
How much of the field of vision is covered by the display? For example, if you hold your mobile phone close to your face its display may cover 20 percent of your field of vision. When the mobile phone is on a table in front of you it may only cover 1 percent. In Figure 7 it might be that the mobile phone covers 2%, the note book covers 10% and the big displays covers 20% of the user's field of vision.

Privacy
If a message is labelled private it shall only be shown to the user. If so, the message will either be presented as a voice message on the user's headset or be shown on the mobile phone display if the user holds the phone in his hands.

Availability of output devices
The EIM application will check what output devices that are available in the system. Which of them is/are within the observable space?

Shall the system always choose the display closest to the user? If this were true, then in the situation above, the system would have chosen the mobile phone.

If no other display is available than the mobile phone, choose the mobile phone if it's in the users hands (object manipulation area).

Display size
What resolution and dpi can the display show? If the message is long, a big display may be better than a small display because it can show more text.

The environment's lighting condition
Direct sunlight may adversely affect the reading experience on a display. If possible, lighting conditions should be able to affect the choice of output device.

Is the environment noisy or silent?
When deciding if the message is to be presented as a voice message on a loudspeaker it is important that the environment is not noisy.

5.6 Explicit input and output devices

The EIM application will monitor explicit input- and output devices to know which of them are available. Information on them will be stored in lists.

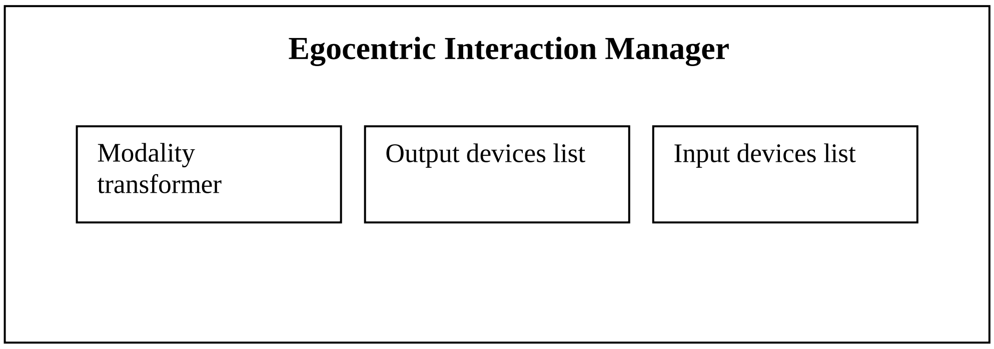


Figure 8: The Egocentric Interaction Manager.

Modality transformer
This is a module within the EIM that makes it possible to transform one modality to another. A textual message can be transformed into a voice message if needed. This module is not implemented in the test application.

Output devices list
Information on available displays and audio devices.

Input devices list
Information on available keyboards and microphones.

5.7 Egocentric Interaction Manager communication

The Egocentric Interaction Manager is a module in the system design of the easyADL project [11]. Below will be given a brief presentation of the different modules in the easyADL's activity-awareness operating system.

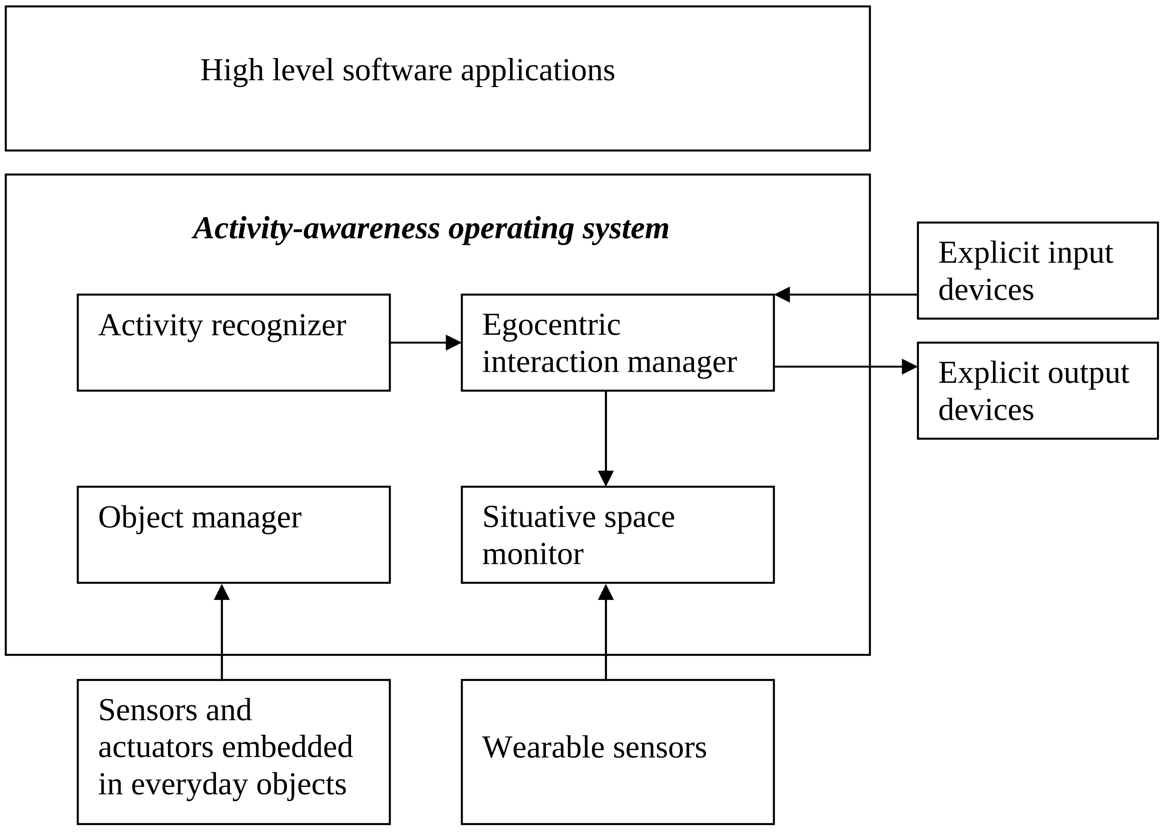

High level software applications
Activity-awareness operating system
Activity recognizer
Egocentric interaction manager
Explicit input devices
Explicit output devices
Object manager
Situative space monitor
Sensors and actuators embedded in everyday objects
Wearable sensors

Figure 9: The activity-awareness operating system adapted from [3].

The **activity-awareness operating system** is a software system running on a wearable computer. It receives and processes data about ongoing human activity from the sensor pool and makes the information available to high-level software applications [11].

The purpose of the **activity recognizer** is to continuously determine the *ongoing activity* and store details about this activity. On request this information shall be provided to other software components.

From a system perspective, when the user changes location, objects come into and leave the *observable space*. The content of the observable and manipulable space indicates the user's intent with the *ongoing activity* [11]. However, a dement person may have forgotten what he/she had intended to do.

The **egocentric interaction manager** is responsible for keeping track of explicit input and output devices. To enable communication between the activity-aware operating system and the user it shall automatically choose the most appropriate input and output device based on the user's physical context.

The **object manager** maintains a real-time model of the environment by storing information about the objects present in the user's environment [3].

The **situative space monitor** stores information about the objects present in the user's environment. The egocentric interaction manager will ask the situative space monitor if a device (explicit input or output) is within the observable space [3].

The modules exchange information by writing data to a common black board. The black board is based on ICE, a middleware for enabling communication over the Internet [13].

5.8 Choosing the best display

In the following picture there are two displays in the user's observable space. Which one of them will best show a message to the user?

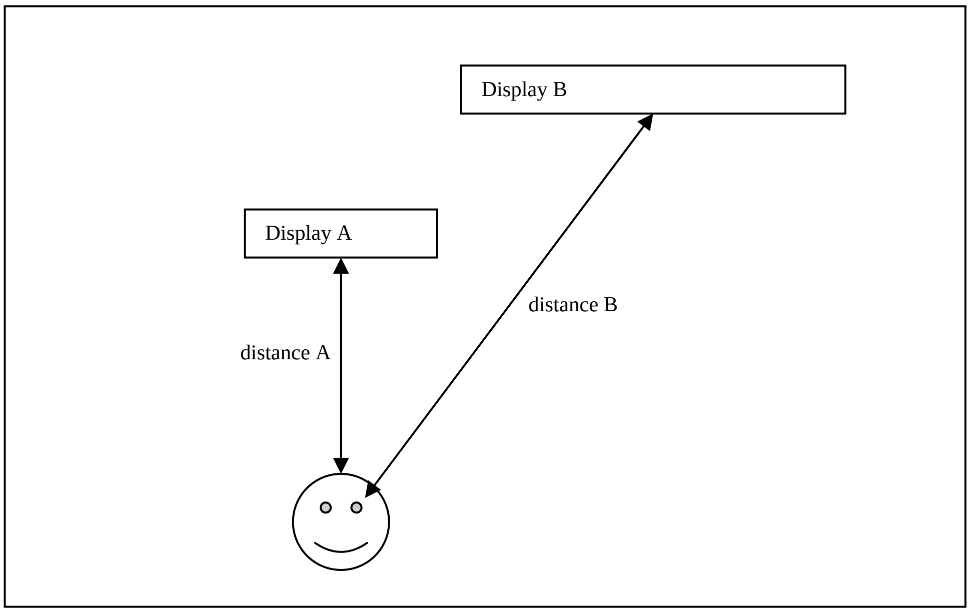


Figure 10: Choosing the best display.

A guideline is to choose the display that covers as much as possible of the user's field of vision:

fA = screen size A/distance A
fB = screen size B/distance B

Example: Display A has screen size 15.4" and is 2 meters away from the user. Display B has screen size 24" and is 3 meters away:

fA = 15.4/2
fB = 24/3

Since fA < fB, display B is the best choice. However, viewing angle needs to be considered too:

fA = (screen size A/distance A) * (1- (wA/90))
fB = (screen size B/distance B) * (1- (wB/90))

Display A is straight in front of the user, wA = 0 degrees.
Display B is viewed at approximately 35 degrees angle, wB = 35.

fA = 15.4/2 * (1- (0/90))
fB = 24/3 * (1- (35/90))

Now the result indicates that display A is the best choice. It is important that the viewing angle is as small as possible, because the bigger the angle is the more difficult it is to view information on the display. Horizontal angle is used in this example and vertical angle could have been considered too. To further refine the selection algorithm, luminance for every display could be considered.

5.9 Egocentric interaction manager functions

The egocentric interaction manager application will offer functionality public to the high level software applications (see Fig. 9). These functions will here be described.

5.9.1 Function parameters

Message
The message that will be presented.

Modality
What form the message will be presented in. It can be a *voice* message (audio), *text* or *image*. A future modality is *tactile feedback*.

Access level
A message can be private, public or broadcasted. If **private**, it is not to be displayed to others than the user. **Public** means that the message will be displayed on devices that the situative space monitor has recognized being in the user’s observable space. **Broadcast** means that it shall be presented on all available devices.

Priority
The message can be replaced by another message with a higher priority at any given time. To make sure that a message can only be removed by an explicit deactivation, give the message high priority.

DeviceID
Identifies which device the message shall be sent to.

MessageID
When calling the function *Activate*, a return parameter is given in the form of a unique number (Integer) that identifies the message. This number is used when deactivating the message.

5.9.2 Functions

Functions will be presented in the following form:
Function name< input parameter 1; input parameter 2>: return value.

The EIM shall offer the following functions towards the high level software applications (see Fig.9):

Activate <message; modality {voice, text, image,vibration};priority {low, medium, high};accesslevel {private, public, broadcast}; deviceID> : integer

Optional parameters: *modality, priority, accesslevel* and *deviceID*. When they are omitted the egocentric interaction manager will treat the message as having default modality, public accesslevel with normal priority. As default the message will be sent to an available output device within the user's observable space.

Deactivate <messageID; deviceID>
Stops a message from being displayed. MessageID specifies which message to deactivate. deviceID is an optional parameter. If unspecified, the message will be removed from all devices that show it.

Inputdevices <set of available inputdevices>
Returns a list of the inputdevices that are registered with the system.

InputdevicesMS <set of available inputdevices within the manipulable space>
Returns a list of inputdevices that are registered with the system and within the manipulable space.

Outputdevices <set of available outputdevices>
Returns a list of outputdevices that are registered within the system.

OutputdevicesOS <set of available outputdevices within the observable space>
Returns a list of outputdevices that are registered within the system and which are within the observable space.

When designing the EIM application it is assumed that only one person at a time will use the system. This has to do with (for this thesis) external factors. The easyADL system will be individually adapted to a specific person, therefore the EIM application is not designed to handle more than one user at a time.

5.10 Algorithms

5.10.1 Select output device

```
1. Check modality:
1.1 Text: go to 5.10.2 Select a display.
1.2 Voice: go to 5.10.4 Select an audio device.
```

5.10.2 Select a display

```
1. Check accesslevel:
   1.1 broadcast: for all displays call 5.10.3 Send message to a display.
   1.2 private or public: go to 2.
2. Is deviceID(:s) specified?
   2.1 yes: for each of the deviceID:s call 5.10.3 Send message to a display.
   2.2 no: go to 3.
3. Iterate the displaylist:
   3.1 Select those that are within the user's observable space
      3.1.1 Has the display the same accesslevel as the message?
          yes: Add the display to a sublist.
4. Is the sublist empty?
   4.1 yes:
        If ((modalitySetByOwner=true) or (switchedModality=true)) then
           4.1.1 switchedModality=false
           4.1.2 Message is added to list of pending messages.
           4.1.3 Notify the sender.
        else
           4.1.4 set modality = voice.
           4.1.5 set switchedModality = true.
           4.1.6 go to 5.10.4 select an audio device.
   4.2 no: Go to 5.

5. Here the field of vision parameter should be calculated as described in
section 5.8 Choosing the best display:

   5.1 fieldOfVision = (screen size/distance) * (1- (viewing angle/90))
   5.2 best (display) = first (display) in the sublist.
   5.3 Iterate the sublist:
      5.3.1 Is (best.fieldOfVision < current.fieldOfVision)?
         5.3.1 yes: best = current.

   5.4 Note: In the software implementation of this algorithm, the first
   display in the subdisplaylist is chosen to be the best.

6. With the best display: go to 5.10.3 Send message to a display
```

5.10.3 Send message to a display

Parameters: message and a display.

```
1. If ( (activeMessage = null) or (activeMessage = message)) then
   1.1 send message to display.
   1.2 activeMessage = message.
   1.3 add displayID to list of devices to which that activeMessage has been sent.
   1.4 Notify owner on message status.
   else go to 2.
2. Is (message.priority > activeMessage.priority)?
   2.1 yes:
      2.1.1 Add active message to list of pending messages.
      2.1.2 Notify message owner on message status (pending).
      2.1.3 activeMessage = message.
      2.1.4 add displayID to list of devices to which that activeMessage has been sent.
      2.1.5 send message to display.
      2.1.6 Notify message owner on message status (active).
   2.2 no:
      2.2.1 Add message to list of pending messages.
      2.2.2 Notify message owner on status (pending).
```

5.10.4 Select an audio device

```
1. Check accesslevel:
   1.1 broadcast: for all loudspeakers call 5.10.5 Send message to an
       audio device.
   1.2 private or public: go to 2.
2. Is deviceID(:s) specified?
   2.1 yes: for each of the deviceID:s call 5.10. 5 Send message to an
       audio device
   2.2 no: go to 3.
3. Iterate the displaylist:
   3.1 Select those that are within the user's observable space
      3.1.1 Has the display the same accesslevel as the message?
          yes: Add the display to a sublist.
4. Is the sublist empty?
   4.1 yes:
        If  ((modalitySetByOwner=true) or (switchedModality=true)) then
           4.1.1 switchedModality=false
           4.1.2 Message is added to list of pending messages.
           4.1.3 Notify the sender.
        else
           4.1.4 set modality = text.
           4.1.5 set switchedModality = true.
           4.1.6 go to 5.10.2 select a display.
   4.2 no: Go to 5.
5. Here the best loudspeaker should have been chosen based on parameters accesslevel, distance etc. However, the first loudspeaker in the sublist is chosen as being the best.
6. With the best loudspeaker: go to 5.10.5 Send message to an audio device.
```

5.10.5 Send message to an audio device

Parameters: message and a loudspeaker.

```
1. If ( (activeMessage = null) or (activeMessage = message)) then
   1.1 send message to loudspeaker.
   1.2 activeMessage = message.
   1.3 add loudspeakerID to list of devices to which that activeMessage has been sent.
   1.4 Notify owner on message status.
   else go to 2.
2. Is (message.priority > activeMessage.priority)?
   2.1 yes:
      2.1.1 Add active message to list of pending messages.
      2.1.2 Notify message owner on message status (pending).
      2.1.3 activeMessage = message.
      2.1.4 add loudspeakerID to list of devices that activeMessage has been sent to.
      2.1.5 send message to loudspeaker.
      2.1.6 Notify message owner on message status (active).
   2.2 no:
      2.2.1 Add message to list of pending messages
      2.2.2 Notify message owner on status (pending).
```

5.10.6 Remove a message

This function removes a textmessage from being displayed.
Required parameter is messagenumber.

```
1. Iterate the list of active messages.
   1.1 Is messagenumber = active.messagenumber?
      1.1.1 yes:  remove message from the deviceID:s that are noted on
      the active message.
2. Is the list of pending messages empty?
   2.1 no: select a message and go to 5.10.1 Select an output device.
```

6. Implementation

6.1. Deciding development environment

Things to consider when deciding what programming language to use for implementing the design:

- The developed software must be able to connect to the blackboard (see description below) for exchanging information with other software.
- What programming languages are used in the system?
- The developed software will run on a mobile device

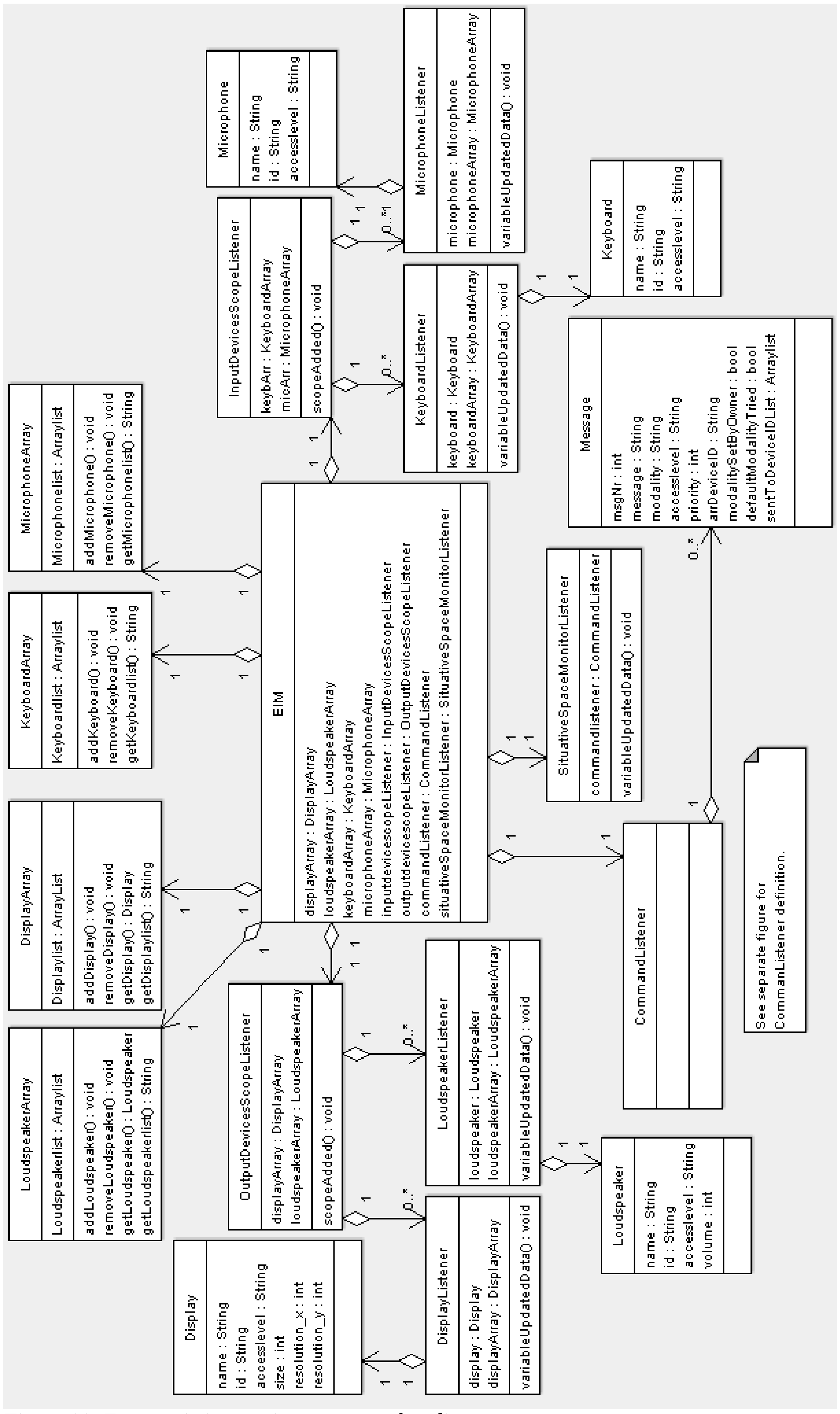
The blackboard is an application based on ICE [7]. ICE clients and servers work together, regardless of the programming language. ICE supports: *C++*, *Java*, *C#*, *Visual Basic*, *Python*, *PHP* and *Ruby*. The black board supports the same programming languages as ICE does.

The activity-aware operating system will be run on a mobile device; for example a laptop, a tablet pc or a mobile phone.

The chosen programming language was C# because it is an object oriented language that is supported by ICE.

6.2. The Egocentric Interaction Manager application

6.2.1 The Egocentric Interaction Manager Class diagram



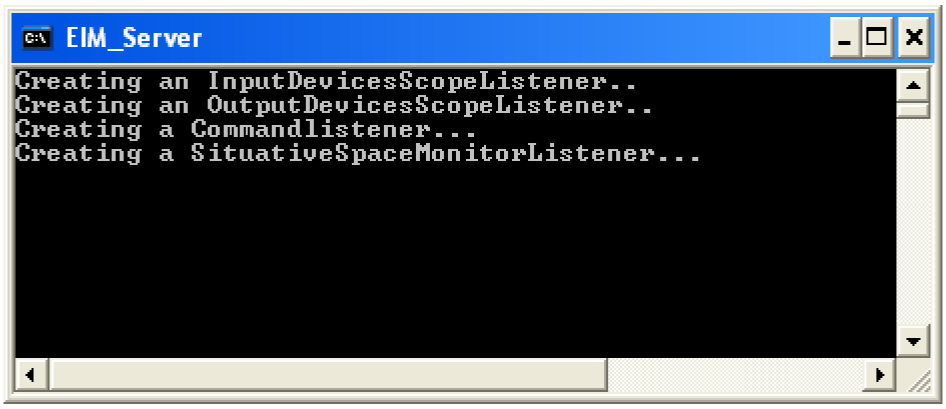
Microphone
name : String
id : String
accesslevel : String
MicrophoneListener
microphone : Microphone
microphoneArray : MicrophoneArray
variableUpdatedData() : void
InputDevicesScopeListener
keybArr : KeyboardArray
micArr : MicrophoneArray
scopeAdded() : void
KeyboardListener
keyboard : Keyboard
keyboardArray : KeyboardArray
variableUpdatedData() : void
Keyboard
name : String
id : String
accesslevel : String
Message
msgNr : int
message : String
modality : String
accesslevel : String
priority : int
arrDeviceID : String
modalitySetByOwner : bool
defaultModalityTried : bool
sentToDeviceIDList : Arraylist
MicrophoneArray
Microphonelist : Arraylist
addMicrophone() : void
removeMicrophone() : void
getMicrophonelist() : String
KeyboardArray
Keyboardlist : Arraylist
addKeyboard() : void
removeKeyboard() : void
getKeyboardlist() : String
EIM
displayArray : DisplayArray
loudspeakerArray : LoudspeakerArray
keyboardArray : KeyboardArray
microphoneArray : MicrophoneArray
inputdevicescopeListener : InputDevicesScopeListener
outputdevicescopeListener : OutputDevicesScopeListener
commandListener : CommandListener
situativeSpaceMonitorListener : SituativeSpaceMonitorListener
SituativeSpaceMonitorListener
commandlistener : CommandListener
variableUpdatedData() : void
DisplayArray
Displaylist : ArrayList
addDisplay() : void
removeDisplay() : void
getDisplay() : Display
getDisplaylist() : String
CommandListener
See separate figure for CommanListener definition.
LoudspeakerArray
Loudspeakerlist : Arraylist
addLoudspeaker() : void
removeLoudspeaker() : void
getLoudspeaker() : Loudspeaker
getLoudspeakerlist() : String
OutputDevicesScopeListener
displayArray : DisplayArray
loudspeakerArray : LoudspeakerArray
scopeAdded() : void
LoudspeakerListener
loudspeaker : Loudspeaker
loudspeakerArray : LoudspeakerArray
variableUpdatedData() : void
Loudspeaker
name : String
id : String
accesslevel : String
volume : int
Display
name : String
id : String
accesslevel : String
size : int
resolution_x : int
resolution_y : int
DisplayListener
display : Display
displayArray : DisplayArray
variableUpdatedData() : void
1
0..*

Figure 11: Egocentric interaction manager classdiagram.

6.2.2 EIM

EIM is the main class in the system. It is responsible for creating workspace on the blackboard area and creating instances of the classes: displayarray, loudspeakerarray, keyboardarray, microphonearray, inputdevicescopelistener, outputdevicescopelistener, commandlistener and situativespacemonitorlistener.

When it instantiates the outputdevicescopelistener, passed arguments are the displayarray and loudspeakerarray. When creating an instance of the inputdevicescopelistener, the keyboardarray and micophonearray are passed as arguments. The commandlistener has the four device arrays as parameters. Finally, the situative space monitor listener takes the commandlistener as argument.



```
Creating an InputDevicesScopeListener..
Creating an OutputDevicesScopeListener..
Creating a Commandlistener...
Creating a SituativeSpaceMonitorListener...
```

Figure 12: The Egocentric Interaction Manager program.

6.2.3 InputDeviceScopeListener

When a keyboard or a microphone adds itself to the blackboard's inputdevice area, the inputdevicescopelistener creates a listener for the device.

6.2.4 OutputDeviceScopeListener

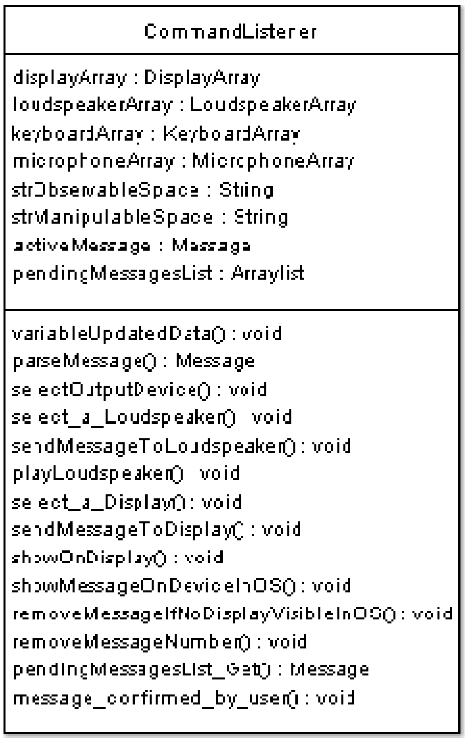
When a display or a loudspeaker adds itself to the blackboards outputdevice area, the outputdevicescopelistener creates a listener for the device.

6.2.5 SituativeSpaceMonitorListener

This class is informed every time the observable space has changed. It forwards this information to the commandlistener.

6.2.6 CommandListener

The CommandListener embodies the core functionality for the EIM application. It listens to commands from high-level applications; see figure 7, and also forwards data from input devices to high-level applications.


CommandListener
displayArray : DisplayArray
loudspeakerArray : LoudspeakerArray
keyboardArray : KeyboardArray
microphoneArray : MicrophoneArray
strObservableSpace : String
strManipulableSpace : String
activeMessage : Message
pendingMessagesList : Arraylist
variableUpdatedData() : void
parseMessage() : Message
selectOutputDevice() : void
select_a_Loudspeaker() : void
sendMessageToLoudspeaker() : void
playLoudspeaker() : void
select_a_Display() : void
sendMessageToDisplay() : void
showOnDisplay() : void
showMessageOnDeviceInOS() : void
removeMessageIfNoDisplayVisibleInOS() : void
removeMessageNumber() : void
pendingMessagesList_Get() : Message
message_confirmed_by_user() : void
Figure 13: The CommandListener.

variableUpdatedData()
The commandlistener listens to four variables: *Activate*, *removeMessageNumber*, *getOutputDevices* and *getInputDevices*.

getOutputDevices: The reply is a list of available output devices.
getInputDevices: The reply is a list of available input devices.
removeMessageNumber: Removes a message from being displayed.
Activate: Selects an output device on which to present the message. It first calls the function *parseMessage*, and then it calls *selectOutputDevice*.

parseMessage()
Parses the message and returns a Message object. If modality is unspecified, default modality will be used.

selectOutputDevice()
If message modality is text, the function *select_a_Display* is called.
If modality is voice, the function *select_a_Loudspeaker* is called.

select_a_Loudspeaker()
Its purpose is to determine if a loudspeaker is within the observable space and to choose the best loudspeaker of those in the observable space, a decision based on the users physical context. If no loudspeaker is found in the observable space, the message modality is changed. If no modality change is possible, the message will be added to a list of pending messages. See flowdiagram below.

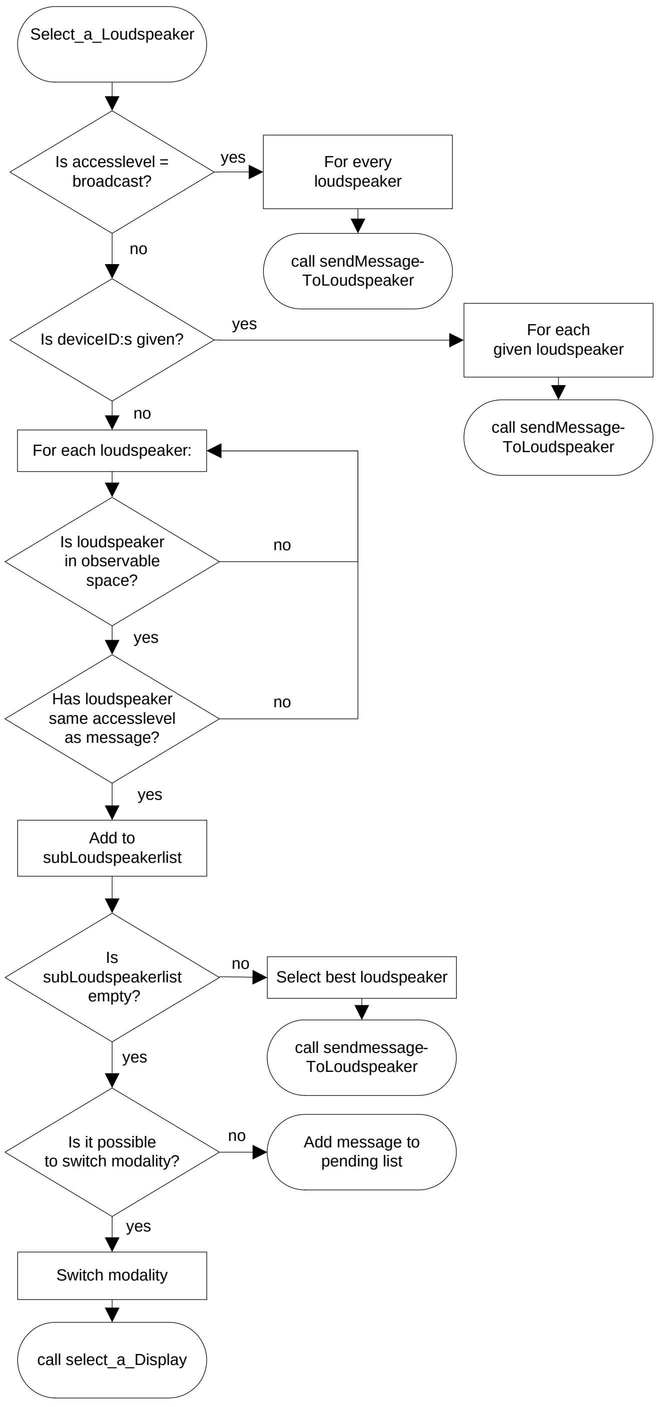

Select_a_Loudspeaker
Is accesslevel = broadcast?
yes
For every loudspeaker
call sendMessage-ToLoudspeaker
no
Is deviceID:s given?
yes
For each given loudspeaker
call sendMessage-ToLoudspeaker
no
For each loudspeaker:
Is loudspeaker in observable space?
no
yes
Has loudspeaker same accesslevel as message?
no
yes
Add to subLoudspeakerlist
Is subLoudspeakerlist empty?
no
Select best loudspeaker
call sendmessage-ToLoudspeaker
yes
Is it possible to switch modality?
no
Add message to pending list
yes
Switch modality
call select_a_Display

Figure 14: Select_a_Loudspeaker flowdiagram.

sendMessageToLoudspeaker()
It checks if any other message is already the active message in the observable space. If so, the message is added to the list of pending messages.

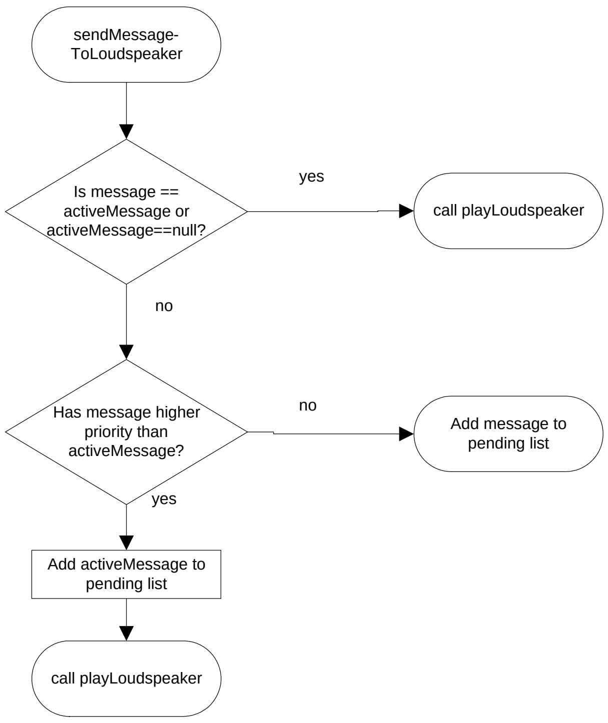

sendMessage-ToLoudspeaker
Is message == activeMessage or activeMessage==null?
yes
call playLoudspeaker
no
Has message higher priority than activeMessage?
no
Add message to pending list
yes
Add activeMessage to pending list
call playLoudspeaker

Figure 15: sendMessageToLoudspeaker flowdiagram.

playLoudspeaker()
Plays the voice message on a given loudspeaker.

select_a_Display()
It determines if displays are within the observable space and chooses the best display (of the displays in the observable space) based on the user's physical context. If no display is within the users observable space, the message modality is changed. If no modality change is possible, the message will be added to a list of pending messages.

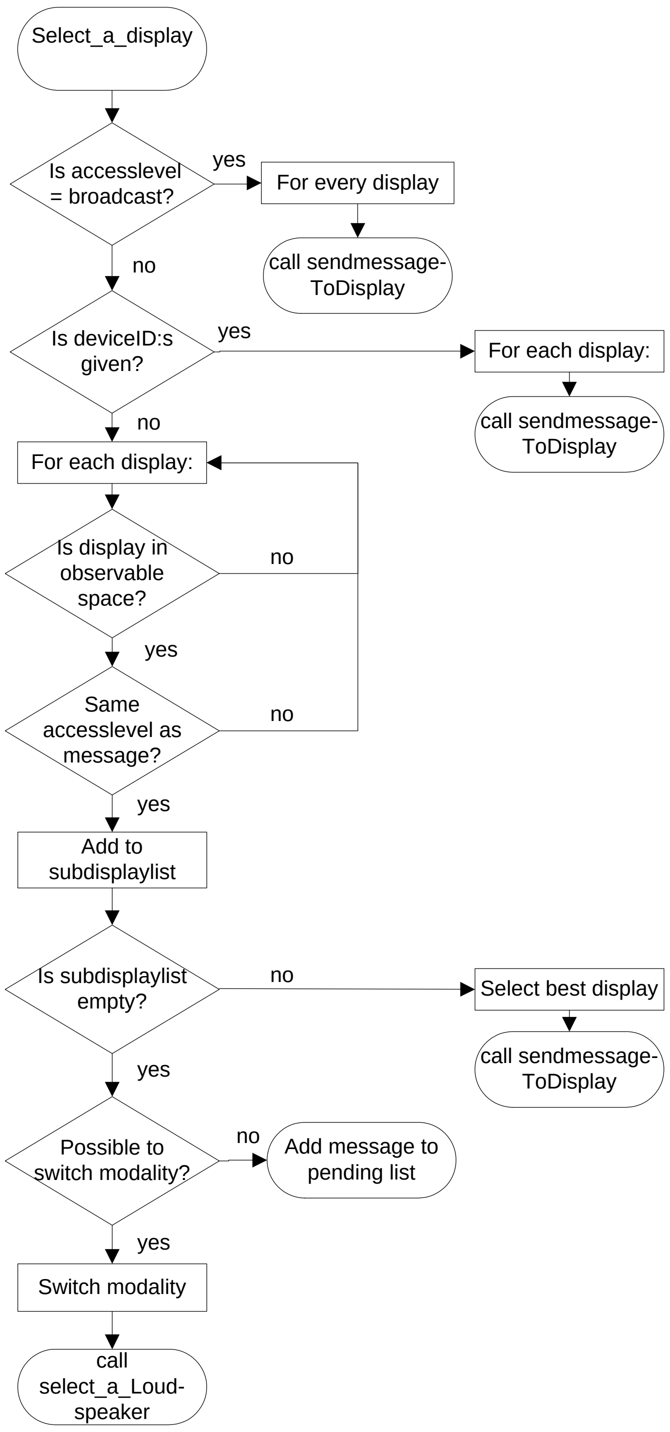

Select_a_display
Is accesslevel = broadcast?
yes
For every display
call sendmessage-ToDisplay
no
Is deviceID:s given?
yes
For each display:
call sendmessage-ToDisplay
no
For each display:
Is display in observable space?
no
yes
Same accesslevel as message?
no
yes
Add to subdisplaylist
Is subdisplaylist empty?
no
Select best display
call sendmessage-ToDisplay
yes
Possible to switch modality?
no
Add message to pending list
yes
Switch modality
call select_a_Loud-speaker

Figure 16: select_a_Display

sendMessageToDisplay()
It checks if any other message is already the active message in the observable space. If so, the message is added to the list of pending messages.

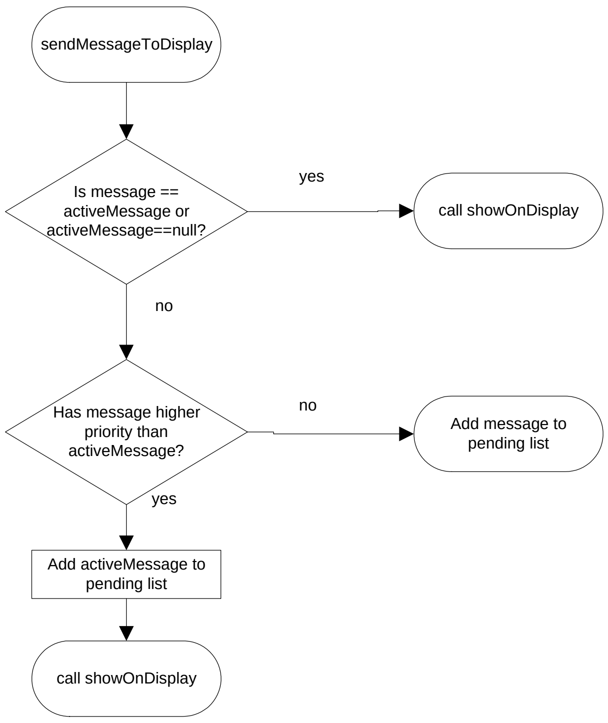

sendMessageToDisplay
Is message == activeMessage or activeMessage==null?
yes
call showOnDisplay
no
Has message higher priority than activeMessage?
no
Add message to pending list
yes
Add activeMessage to pending list
call showOnDisplay

Figure 17: sendMessageToDisplay

showOnDisplay()
Shows a message on a given display.

removeMessageNumber()
Stops a given message from being displayed.

showMessageOnDeviceInOS()
The function is called every time the user’s observable space is changed. The call is made from a function in the SituativeSpaceMonitorListener class. Its purpose is to show a message on an output device in the user’s observable space.

removeMessageIfNoDisplayVisibleInOS()
This function is called each time the user’s observable space is changed. It removes a message from a display if no display is visible in the users observable space.

pendingMessagesList_Get()
returns a message with the highest priority and lowest possible messagenumber. The pendingmessageslist is unsorted but this function makes the reply sorted. Sort order is: 1 Priority. 2. Messagenumber.

message_confirmed_by_user()
It checks if the device that initiated the call to this function is within the user's manipulable space. If so, the active message is removed from being displayed.

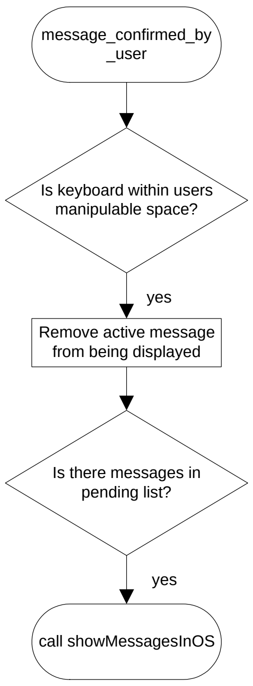

message_confirmed_by_user
Is keyboard within users manipulable space?
yes
Remove active message from being displayed
Is there messages in pending list?
yes
call showMessagesInOS

Figure 18: message_confirmed_by_user

6.2.7 DisplayArray

This class offers functionality to add, remove and get a Display and to get a list of Displays.

6.2.8 LoudspeakerArray

Has functionality to add, remove and get a Loudspeaker and to get a list of Loudspeakers registered in the system.

6.2.9 KeyboardArray

This class offers functionality to add, remove and get a Keyboard and to get a list of Keyboards.

6.2.10 MicrophoneArray

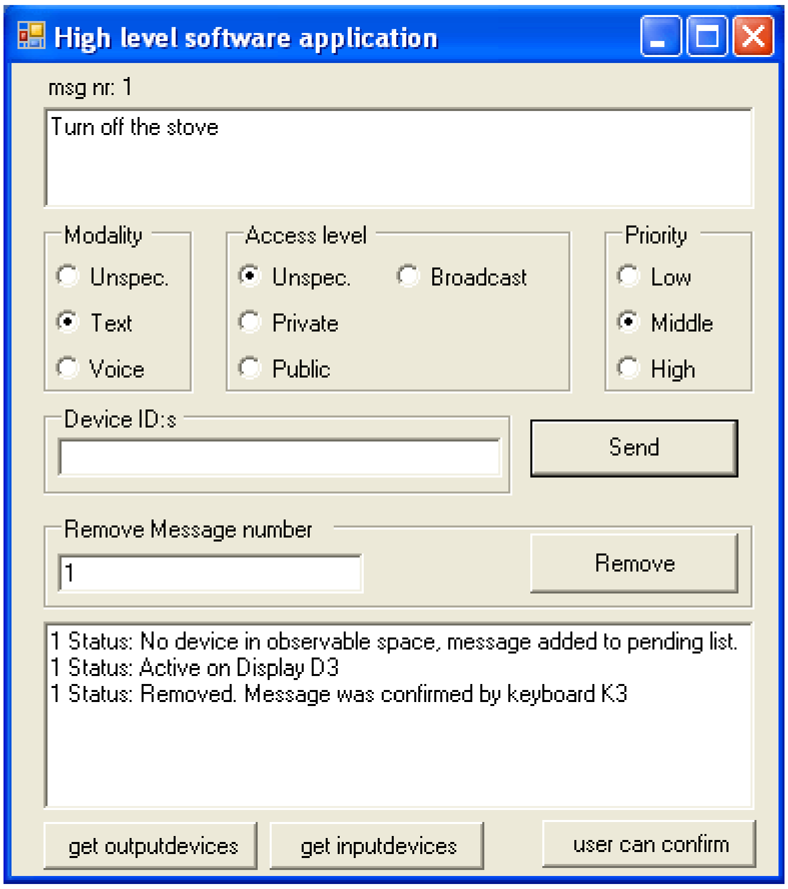
Has functionality to add, remove and get a Microphone and to get a list of Microphones.

6.3 Test applications

In order to evaluate the egocentric interaction manager some other software applications have been implemented.

6.3.1 High level software application

This application can send messages to the user. The message is sent to the egocentric interaction manager that decides how and when it shall be presented to the user. For a message the modality, access level, priority and deviceID:s can be specified.


High level software application

msg nr: 1

Turn off the stove

Modality: Unspec. [✓ Text] Voice

Access level: [✓ Unspec.] Broadcast Private Public

Priority: Low [✓ Middle] High

Device ID:s:

Remove Message number: 1

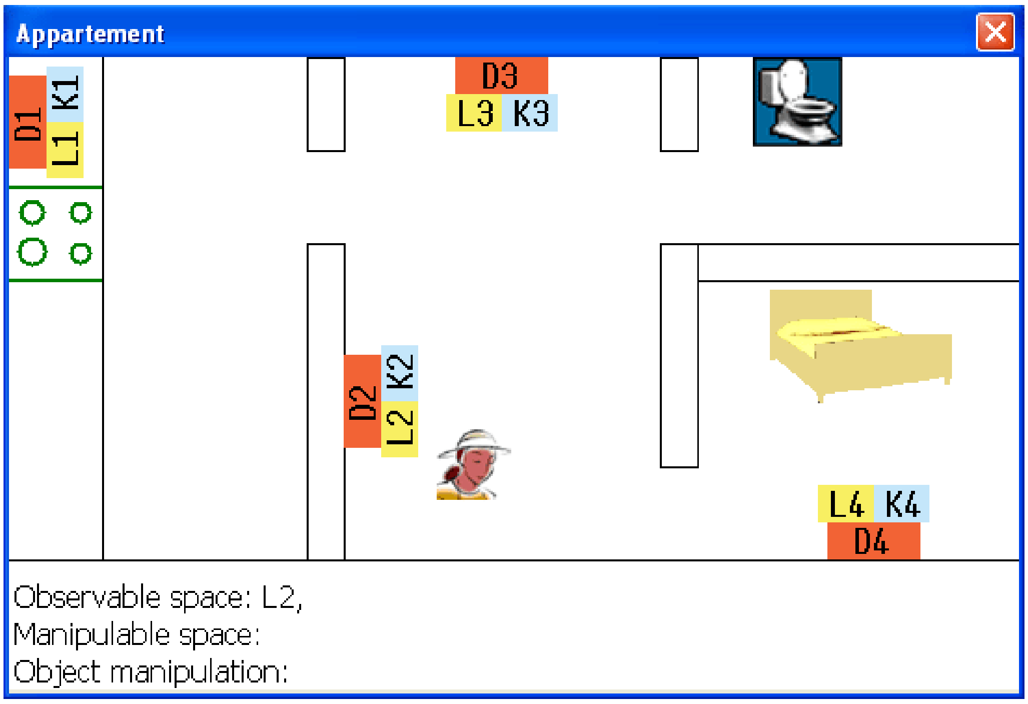
```
1 Status: No device in observable space, message added to pending list.
1 Status: Active on Display D3
1 Status: Removed. Message was confirmed by keyboard K3
```

Figure 19: The message and its status are shown in above figure.

For each message feedback will be given. In the figure above, it begins with the information that no device (display or loudspeaker) is found in the observable space so the message is added to a list of pending messages. Next, a device is found in the observable space and the message is presented on loudspeaker L1. Finally the application is noted that the message is removed.

6.3.2 Situative space monitor

The application shows a user apartment with in- and output devices. It keeps track of which devices are within the observable space, manipulable space and object manipulation.



Observable space: L2,
Manipulable space:
Object manipulation:

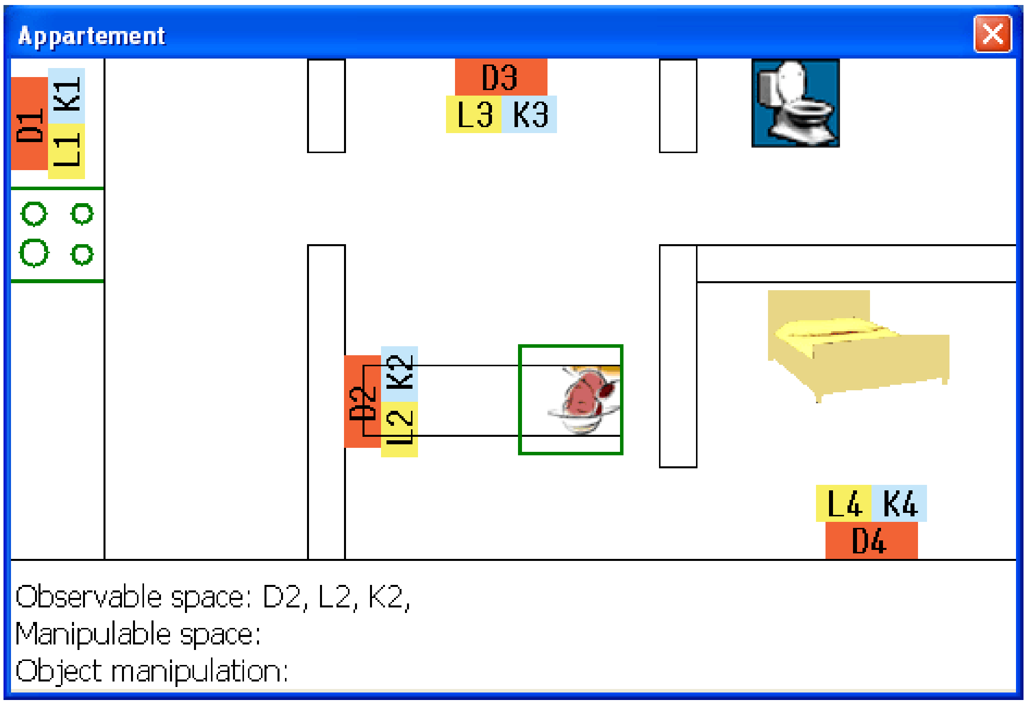
Figure 20: Situative space monitor application.

In the kitchen there is a microwave oven combined with a display D1, loudspeaker L1 and keyboard K1. In the living room there is a computer (D3, L3, K3) and a flat screen tv (D2, L2, K2). In the bedroom there is a small tv (D4,L4,K4).

The user can carry a mobile phone (D5, L5, K5, M5). The term, M5, stands for the microphone. In figure 20 loudspeaker L2 is within the user's observable space so a message with unspecified modality or modality set to voice can be presented on that device.

Each time the observable space has changed, the situative space monitor sends this information to the egocentric interaction manager. The egocentric interaction manager always has an up to date picture of what devices are within the user's observable space.

In figure 20 the woman icon illustrates the user. She can be moved around in the apartment by touching the arrow keys on the keyboard.

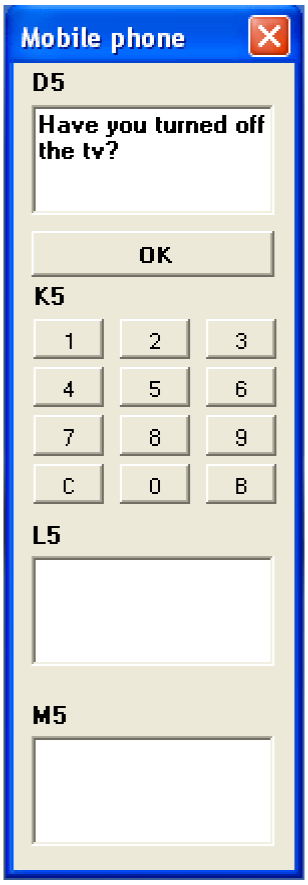


Observable space: D2, L2, K2,
Manipulable space:
Object manipulation:

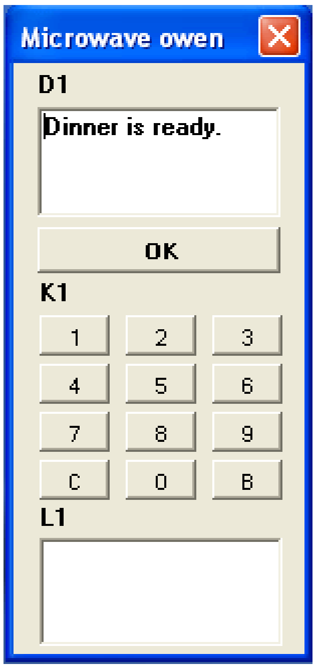
Figure 21: Observable and manipulable space illustrated by rectangles around the user icon.

The observable and manipulable spaces are illustrated in figure 21. The rectangle in the figure from the user Mary to the display D2 illustrates *the observable space*. The rectangle surrounding the user is *the manipulable space*.

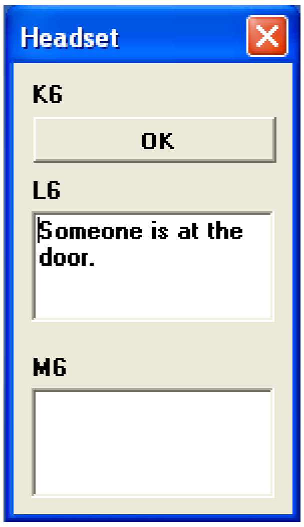
6.3.3 Client applications



Have you turned off the tv?



Dinner is ready.



Someone is at the door.

Figure 22A: Mobile phone *22B: Microwave oven.* *22C: Headset.*

Illustrated above are three client applications. They get their properties from a textfile which makes it easy to alter the system by creating new clients or changing the existing client's properties.

The data order in the textfile for the Microwave oven program:
Clienttype, displayID, displayAccesslevel, displaySize, displayResolutionX, displayResolutionY, loudspeakerID, loudspeakerAccesslevel, loudspeakerVolume, keyboardID, keyboardAccesslevel.

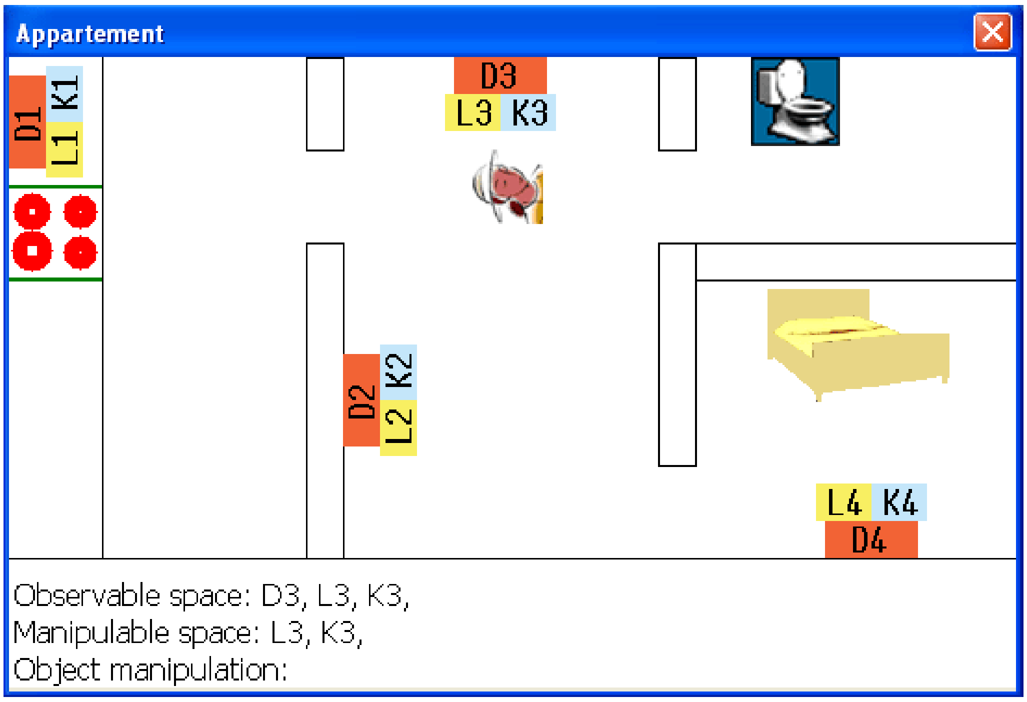
The textfile:
```
Microwave oven
D1
public
10
320
240
L1
public
10
K1
public
```

The mobile phone application is registered in the system as four separate devices: loudspeaker, display, keyboard and a microphone. When the button

"OK" is pressed, a confirmation message is sent to the system that a message is acknowledged by the user.

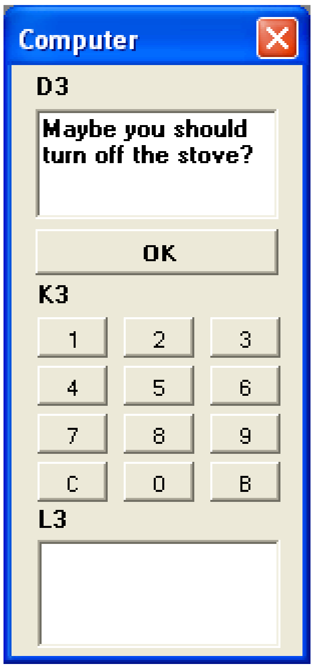
6.4 The system in action

Here is a scenario in which the user, Mary, has cooked some food on the stove. When the cooking is finished she decides to check her email. However, she has forgotten to turn off the stove so the system sends out a notification to inform her that the stove is on.



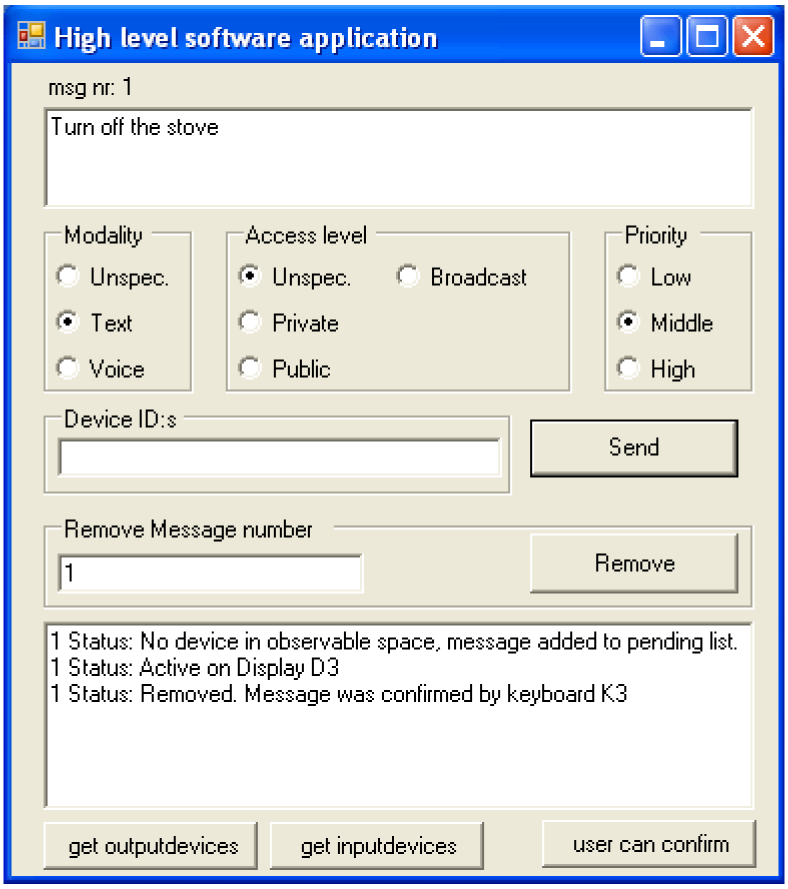
Observable space: D3, L3, K3,
Manipulable space: L3, K3,
Object manipulation:

Figure 23: The user, Mary, in front of her computer.



Maybe you should turn off the stove?

Figure 24: A message is shown on the computer screen.



msg nr: 1

Turn off the stove

Modality: Unspec. [✓ Text] Voice

Access level: [✓ Unspec.] Broadcast Private Public

Priority: Low [✓ Middle] High

Remove Message number: 1

```
1 Status: No device in observable space, message added to pending list.
1 Status: Active on Display D3
1 Status: Removed. Message was confirmed by keyboard K3
```

Figure 25: Message status illustrated.

When Mary has read the message, she confirms it by pressing the OK-button on the computer's keyboard.

In figure 25, message status is given. First the message is shown on display D3. Then it is confirmed by the usage of keyboard K3 so the message is removed from being displayed.

If the stove is not turned off within a reasonable amount of time, ideally the system will show the message again. This functionality belongs to the high level software application and is not implemented in this version.

7. Evaluation

7.1 Hypotheses

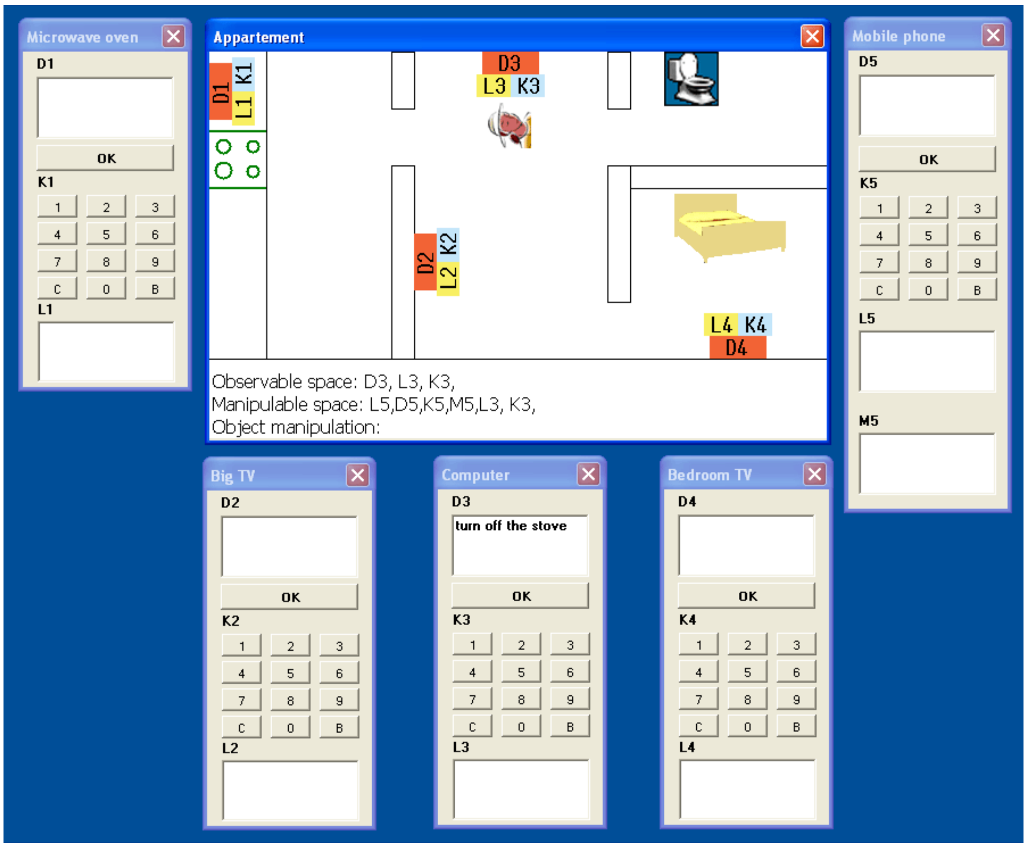
Users will prefer a dynamic selection of output device before broadcasting (see version 1 and 2 on page 41-42). They will prefer using a confirmation button to clear out messages rather than letting the system clear out messages. Users will prefer voice messages before text messages.

7.2 Experimental setup

In order to evaluate the Egocentric Interaction Manager, six persons have tested the simulation environment. Afterwards they answered a questionnaire on how they perceived the system and its way of delivering information.

Three of the subjects were approximately 80 years old; the other three were 40 years old. There were two women and four men, none of them had dementia. Before the test they were informed that the system is meant as a tool for people with mild dementia to help them perform activities in their daily lives. Each interview took about half an hour to do. The tests were done in July 2008.

The subjects were shown how to use the system to move the user, *Mary*, by using the arrow keys on the keyboard. The stove can be turned on and off by a mouse click, the toilet is used/ flushed by a mouse click, too.

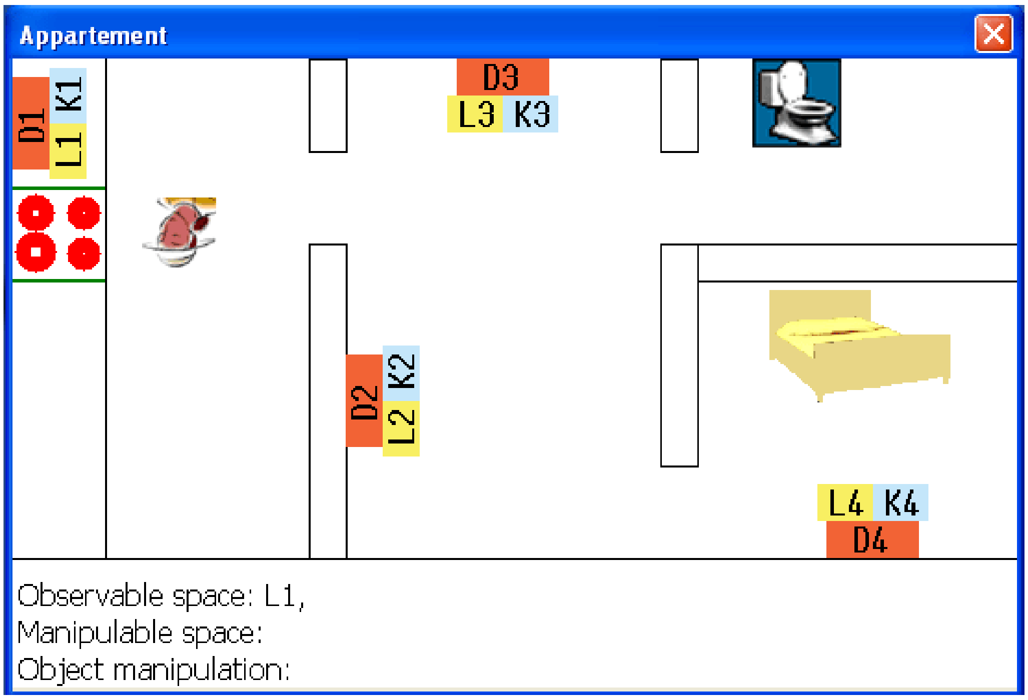


Observable space: D3, L3, K3,
Manipulable space: L5,D5,K5,M5,L3, K3,
Object manipulation:

Figure 26: The simulation environment.

The *Wizard of Oz* technique was used for simulation [13]. The subjects interacted with a computer system which they thought was autonomous, but which was actually being operated or partially operated by an unseen human being. Messages were sent by the test coordinator from a computer connected to the system that the user was not aware of.

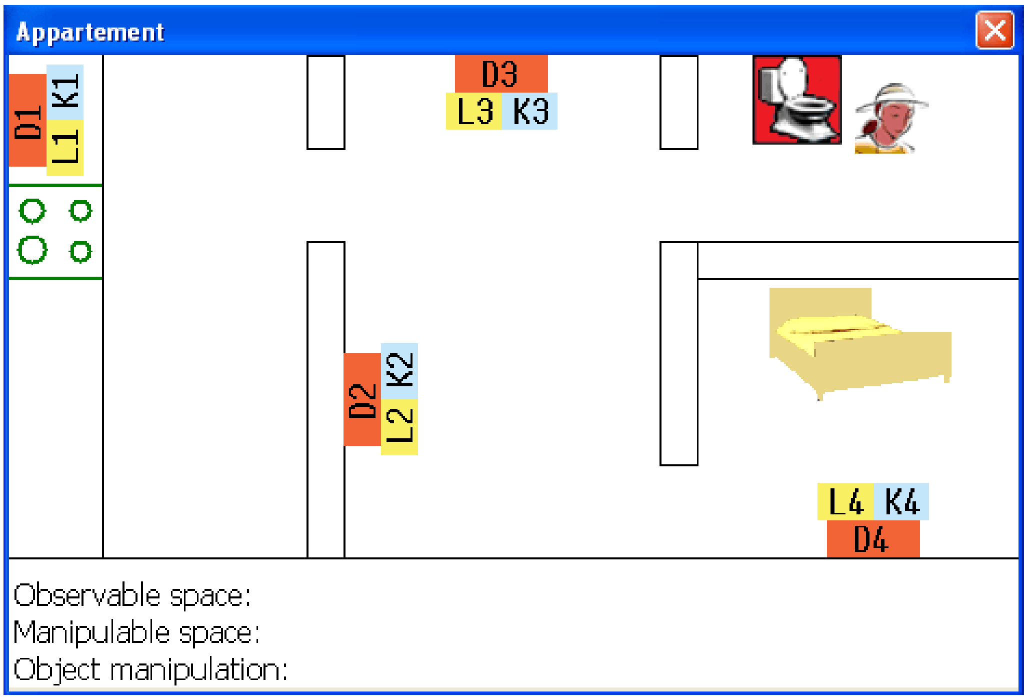
Scenario 1:
The user, Mary, walks into the kitchen to make some lunch. She decides to make a soup. She turns on the stove and cooks the soup on the stove. When the soup is ready Mary takes it off the stove and walks into the living room where she enjoys her meal in front of the television set. However, she has forgotten to turn off the stove and the system will instruct her to turn it off.



Observable space: L1,
Manipulable space:
Object manipulation:

Figure 27: The stove changes colour to red when it is turned on. .

Scenario 2:
After the meal Mary needs to go to the toilet. She walks into the bathroom. When she is done she forgets to flush the toilet and needs to be reminded to do so.



Observable space:
Manipulable space:
Object manipulation:

Figure 28: The toilet background colour switches from blue to red when it needs to be flushed.

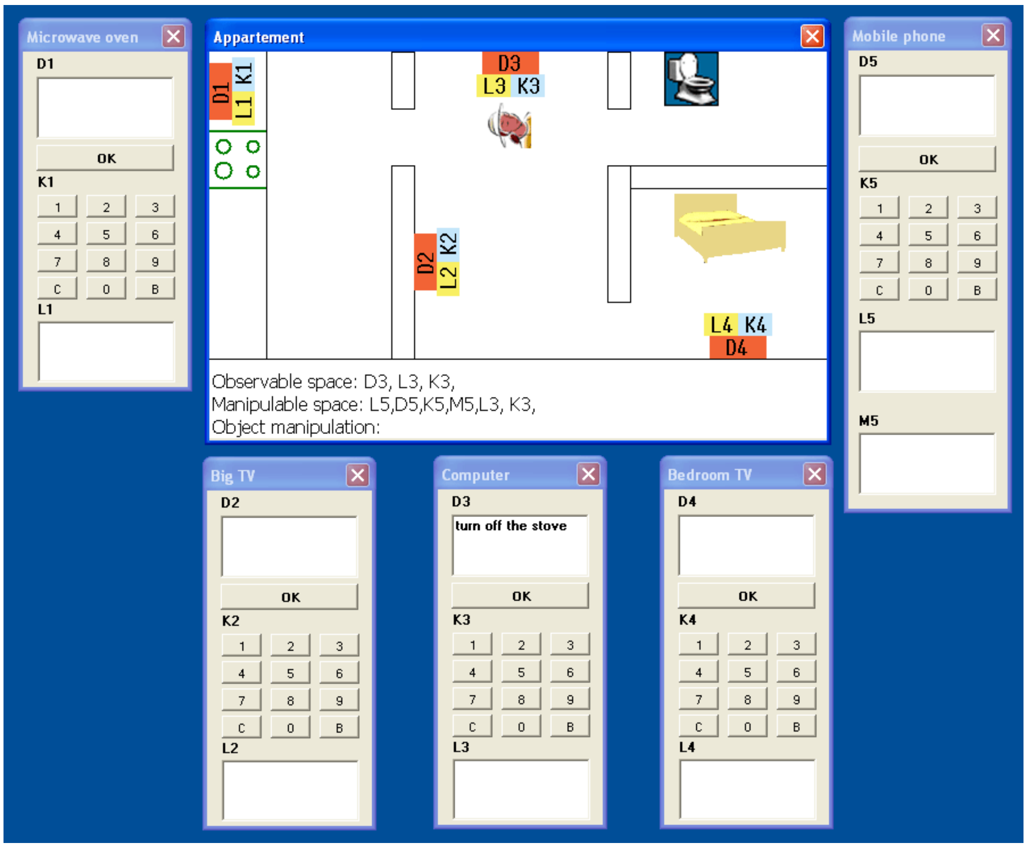
The experimental design was within-subjects where each user performed under each different condition [13]. The independent variables are presented in Table 1.

	Confirmation by the user is requested by the system	**Message display mode**
Version 1	yes	Dynamic presentation depending on observable space
Version 2	no	Broadcast

Table 1: test bed differences.

Version 1:
To display a message an output device in the user's observable space will be chosen. If no output device is available, the message will be queued.

The user is assumed to confirm a message by pressing a confirmation button on a device in his/her manipulable space.



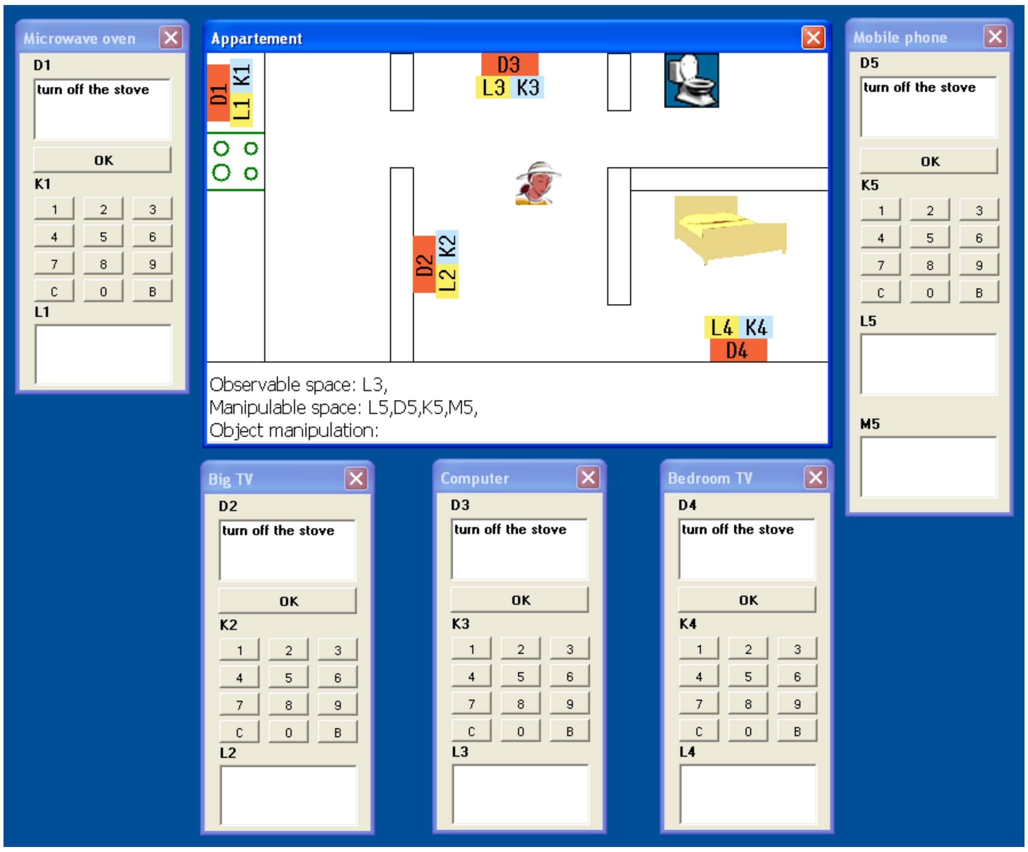
Observable space: D3, L3, K3,
Manipulable space: L5,D5,K5,M5,L3, K3,
Object manipulation:

turn off the stove

Figure 29: Version 1: Messages are shown on a device in the user's observable space.

Version 2:
A message will be broadcasted to all available output devices regardless of access level (private / public) or if the device is within the user's observable space.

The user can not confirm messages; the system itself will decide when a message shall be removed.



Observable space: L3,
Manipulable space: L5,D5,K5,M5,
Object manipulation:

Figure 30: Version 2: Messages are shown on all devices in the system.

At the end of the experiment the users listened to voice messages played through a computer loudspeaker and a Bluetooth headset. The message "turn off the stove" was played to exhibit voice message mode.

7.3 Analysis

A full version of the results is available in Appendix A. Due to the limited number of subjects no quantitative analysis could be made.

Which version did the subjects prefer to use for getting information on a display? Four of the six subjects preferred the broadcast version (version 2). "*It is safer if a message is shown everywhere*" was one motivation. Another subject thought that people with dementia live with a family or have an assistant that can help them perform the requested activities.

Shall a message be cleared by the user pressing a confirmation button, or shall the system itself decide when to clear a message?
Five of six subjects wanted the system to clear messages. "*Will a dement person remember to press a confirmation button?*", "*It is unnecessary to press a confirmation button*".

Shall a message be shown directly on the television screen or on a separate display above the television set?
Four of six thought they would prefer to have messages shown directly on the television screen. *"I will see it directly when watching TV", "It is easy to miss if it was shown on a separate display".*

What message modality would be best in a home environment (text on a display or voice messages)?
Four of six subjects preferred voice messages. *"I don't look on a display when I do stuff."*

Note that because of the limited number of subjects in this pilot study, it is difficult to statistically confirm or reject the hypotheses (see section 7.1).

The subjects preferred messages which were broadcasted rather than those presented on a device in the user's observable space. Voice messages were preferred to text messages. Two of the subjects suggested that an audio signal could be used to catch the user's attention in combination with a text message presented on a display near the subject

8. Conclusion

This chapter presents the results from this master’s thesis based on the goals in section 2 and the evaluation of the system in section 7.3.

The first goal was to implement a simulation environment that makes use of the situative space model. Both the *situative space monitor* and the *egocentric interaction manager* make use of the situative space model. When in- and output devices come into the user’s observable and manipulable space, the *situative space monitor* informs the *egocentric interaction manager* about it.

The second goal was to design a software component that selects the best modality and best input and output device for the communication between a software application and the user, based on the user’s physical context. The design for this is outlined in section 5.

A prototype that exhibits parts of the design was implemented as the *egocentric interaction manager* application (fulfilment of goal number three).

To run the prototype some other software modules needed to be in the system. The following modules were implemented: a high-level software application, a situative space monitor combined with a simulated user environment and some client applications with input and output devices. This fulfilled goals numbered four, five, six and seven.

The prototype was evaluated in a pilot study where two versions of the prototype were tested by six persons. The subjects preferred messages being broadcasted rather than displayed on one device at a time. This made sure that the message was noticed by the user.

Voice messages were preferred to text messages. An ideal solution might be an audio signal to catch the user’s attention combined with the text message shown on displays, an idea raised by two of the six subjects.

9. Future work

A logical next step for the egocentric interaction manager would be to include a text-to-speech module.

A nice concept would be to play a specific sound before a voice message is played. It should ideally catch the user's attention and make him/her prepared to hear a message, something similar to the "bell signal" at a railway station before a train is announced.

The usage of movements in displays to attract attention would also be a nice feature to add. As for now, the simulation environment is very basic and could be enhanced.

10. Acknowledgements

I would like to thank my supervisor Thomas Pederson for giving discussions and guidance during the progress of this thesis. Thanks also to Daniel Sjölie for his assistance with using ICE communications platform. Finally, thanks to Barbara Frankel for her proofreading.

11. References

[1]. easyADL - Independent life despite dementia. Website http://www.cs.umu.se/research/easyadl/, accessed 2008-02-21

[2] Backman, A., Bodin, K., Bucht, G., Janlert, L.-E., Maxhall, M., Pederson, T., Sjölie, D., Sondell, B., & Surie, D. (2006). easyADL – Wearable Support System for Independent Life despite Dementia. Workshop on Designing Technology for People with Cognitive Impairments, CHI2006, April 22-23, Montréal, Canada.

[3] Surie, D., Pederson, T. An Activity-Centered Wearable Computing Infrastructure for Intelligent Environment Applications. In Proceedings of IFIP EUC 2007 Conference on Embedded and Ubiquitous Computing, Springer LNCS 4808, pp. 456-465.

[4] Tara Matthews, Anind K. Dey, Jennifer Mankoff, Scott Carter, Tye Rattenbury. A Toolkit for Managing User Attention in Peripheral Displays, in *17th annual ACM symposium on User interface software and technology* (2004) ISBN: 1-58113-957-8

[5] Gaver, W. T. *et al.* Effective sounds in complex systems: the ARKola simulation. (In *Proc of CHI '91*, pp.85-90.) http://portal.acm.org/citation.cfm?doid=108844.108857

[6] Noha Ibrahim, Frédéric Le Mouël, Stéphane Frénot. C-ANIS – A Contextual, Automatic and Dynamic Service-Oriented Integration Framework. In *Ubiquitous Computing Systems UCS 2007*, pp. 118-133.

[7] Jin Wook Lee, Su Myeon Kim, Hun Lim, Mario Schuster, Alexander Domene. A Software Architecture for Virtual Device Composition and Its Applications. In *Ubiquitous Computing Systems UCS 2007*, pp. 150-157.

[8] Surfa utan hinder - Funktionshinder. Website. http://www.hi.se/surfautanhinder/funktionshinder_kognition.htm, accessed 2008-02-26.

[9] Sjöblom,S., Zingmark, S. A User Interface Proposal for a Wearable Cognitive Prosthesis Supporting Dementia Patients with Activities of Daily Living. http://www.cs.umu.se/education/examina/Rapporter/SjoblomZingmark.pdf, accessed 2008-03-03.

[10] Surie, D., Pederson, T., Lagriffoul, F., Janlert, L.-E., & Sjölie, D. (2007). Activity Recognition using an Egocentric Perspective of Everyday Objects. In Proceedings of IFIP UIC 2007 Conference on Ubiquitous and Intelligent Computing, Springer LNCS 4611, pp. 246-257.

[11] Surie, D., Lagriffoul, F., Pederson, T., Sjölie, D. (2007). Activity Recognition based on Intra and Extra Manipulation of Everyday Objects. In Proceedings of IFIP UCS 2007 Conference on Ubiquitous Computing Systems, Springer LNCS 4836, pp. 196-210.

[12] The Internet Communcations Engine (ICE). Website. http://www.zeroc.com/ice.html, accessed 2008-01-21

[13] Dix, A., Finlay, J., Abowd, G.D., Beale, R. Human-Computer Interaction. Pearson Prentice Hall 2004.

Appendix A – Study with the six persons that testran the simulation environment (in Swedish)

För detaljer om hur test och intervjuer har gjorts, se kapitel 7 Evaluation i rapporten. Tre pensionärer har testat systemet, deras svar markeras som: (1), (2), (3). De tre andra som testat systemet är i yrkesverksam ålder, deras svar markeras med: (4), (5), (6).

Ringa in det system som testades först:	V1	V2
	(2),(4),(6)	(1),(3),(5)

1. Vilket system skulle du föredra att använda av de två som visar meddelanden på bildskärm:

☐ V1: Meddelandet visas på en bildskärm i taget (när jag ser på den).

☐ V2: Meddelandet visas på alla bildskärmar samtidigt.

V1: (3),(6)
Mindre förvirrande att se på en bildskärm i taget (3).
Det borde räcka (6).

V2: (1),(2),(4),(5)
Det känns säkrare om ett meddelande visas överallt. Att glömma att stänga av spisen är farligare än att glömma att spola toaletten (1).
Man kan inte, även om man koncentrerar sig på en skärm, missa meddelandet.(2)
Ofta bor en senildement fortfarande med familj eller har en assistent. De kan bistå med att stänga av spis etc.(4)
Då vissa situationer är farliga (en spis som inte stängts av) är det lämpligast att visa på alla. Olika nivåer kan vara en förbättring (5).

2. Hur ska ett meddelande bekräftas/ tas bort från en bildskärm?

☐ V1: Användaren bekräftar meddelandet via en OK-knapp.

☐ V2: Systemet tar själv bort meddelandet när det inte behövs visas längre.

V1: (2)
Jag vidtar en aktiv åtgärd och ser genast resultatet (2)

V2: (1),(3),(4),(5),(6)
Kommer en dement ihåg att använda OK-knappen? (1)
Mer motiverat att förflytta sig och utföra momentet som meddelandet upplyser om (3).
En senildement kanske inte kommer ihåg att han har tryckt på OK-knappen (4).
En OK-knapp är inte detsamma som att man stängt av spisen (5).
Ett onödigt moment att trycka OK (6)

3. **Ska ett meddelande visas direkt på tv-skärmen eller på en separat skärm ovanför tv:n?**

☐ a) Direkt på tv-skärmen.

☐ b) På en separat skärm ovanför tv:n

a): (3),(4),(5),(6)
Om jag tittar på tv:n så ser jag det direkt (3)
För att man är fokuserad på tv-skärmen. Kanske kan det göras så att man ser på tv men man noterar också när ett meddelande visas på en separat skärm? (4)
Det är tv:n som man naturligt ser på. Det är dumt att tillföra nya skärmar (5).
Man kan lätt missa det på en separat skärm (6).

b): (1),(2)
Inte blanda in tv:n. Kombinera gärna ett textmeddelande med en ljudsignal, exempelvis ”pling-plong” (1).
Då stör den inte den bildsekvens som jag tittar på (2)

4. Om du bara får ha ett sätt att ta emot meddelanden (ljud eller bild). Vilket väljer du?

☐ a) Text meddelande som visas på bildskärm

☐ b) Ljud meddelande via en högtalare eller headset

a) Text: (4),(6)
Det är lättare att komma ihåg ett meddelande i textform. Man kanske glömmer sitt headset som senildement? (4)
Det kan vara svårt att uppmärksamma ett ljudmeddelande om man hör dåligt eller att andra ljudkällor stör, exempelvis att man lyssnar på musik. (6)

b) Ljud: (1),(2),(3),(5)
Inte ser jag på en bildskärm när jag gör något? En idé kan ju vara att det plingar till och då söker man upp en bildskärm för att ta del av ett meddelande (1).
Jag föredrar ljudmeddelande. Vid dålig hörsel skulle jag ha valt textmeddelande, vid nedsatt synförmåga är ljudmeddelande att föredra (2).
Om man är förvirrad är nog ljud lättare att ta till sig (3).
Att se text kräver egen handling – ljud är mer direkt (går inte att bomma). Det bästa vore en kombination; ett ljud som får en att titta på skärmen. Ljud/tal i örat hos en lätt förvirrad person kan vara mer förvirrat trots allt (5).