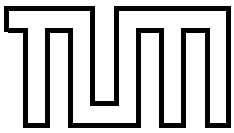
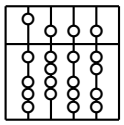


Technische Universität München
Fakultät für Informatik

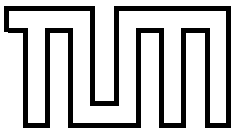


Diplomarbeit

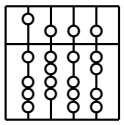
Usability Engineering for Ubiquitous Computing

ARCHIE: Augmented Reality Collaborative Home Improvement Environment

Christian Kulas



Technische Universität München
Fakultät für Informatik



Diplomarbeit

Usability Engineering for Ubiquitous Computing

ARCHIE: Augmented Reality Collaborative Home Improvement Environment

Christian Kulas

Aufgabensteller: Prof. Dr. Bernd Brügge

Supervisor: Dipl.-Inform. Christian Sandor

Date: 15. Juni 2003

Ich versichere, dass ich diese Diplomarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. Juni 2003 Christian Kulas

Zusammenfassung

Technologien wie Wearable Computing haben die Vision von überall platzierten, unauffälligen Computern durch Integration in die Kleidung. Augmented Reality reichert die Sinne des Benutzers, mit digitalen Informationen eingeblendet in die Realität, an. Ubiquitous Computing zielt ab auf die Kreation von vielen intelligenten Geräten, die in das normale Leben eingebettet werden, bis sie nicht mehr davon unterscheidbar sind. Die Mixtur dieser neuen Technologien stellt neue Herausforderungen an das Benutzbarkeits-Engineering.

Diese Diplomarbeit stellt ein passenden Framework für Benutzbarkeits-Studien vor, wobei sowohl Prozess- wie auch Software-Fragen beleuchtet und behandelt werden.

Eine automatische Lösung zur Speicherung von Messdaten wird präsentiert, wie auch ein Werkzeug für die schnelle manuelle Notiznahme. Alle Leistungs-Messungen können schließlich mit einer flexiblen Interpreter-Lösung auf Skriptbasis visualisiert werden.

Zur Verifizierung des vorgestellten Frameworks wurde eine Benutzbarkeits-Studie, die zwei verschiedene Menü-Typen mit tragbaren Eingabegeräten vergleicht, durchgeführt.

Abstract

Technologies like Wearable Computing have the vision of non-obtrusive computers everywhere by integrating them into clothes. Augmented Reality enhances the user's senses with digital information augmented into reality. Ubiquitous Computing aims at creating many small intelligent devices weaving them into the fabric of everyday life until they are indistinguishable from it. The mixture of these new technologies poses new problems to usability engineering.

This thesis proposes a matching framework for usability evaluations, covering both process as well as software issues.

A fully automated logging solution is presented, as well as a manual data entry tool for quick note taking. All performance measurements can finally be visualized with a flexible interpreter solution based on scripts.

A study comparing two different menu types with wearable input devices was conducted to validate the proposed framework.

Preface

Acknowledgments I would like to thank my supervisor Christian Sandor for his help and advice concerning this thesis.

I am indebted to Prof. Bernd Brügge and Prof. Gudrun Klinker for making this thesis possible and giving helpful remarks about my work.

Without the combined efforts of all ARCHIE team members, Otmar Hilliges, Manja Kurzak, Felix Löw, Marcus Tönnis, Franz Strasser, Johannes Wöhler, and Bernhard Zaun and the supportive staff including Martin Bauer, Asa MacWilliams and Martin Wagner, and all other members of this chair who got involved in our work, the ARCHIE project and this thesis would not have been possible.

Thanks to all voluntary participants who agreed to take part in my sample usability study.

Finally I would like to thank my family and friends for the emotional support.

Overview

Contents

List of Figures

Part I

Foundation

1 Introduction

Why do we need usability engineering? Scope and structure of my thesis.

1.1 Motivation

Human-Computer-Interfaces have advanced in huge steps the last couple of decades. We went from stamping holes in cardboards which were fed to batch processing huge stationary computers, to augmenting our reality with mobile devices. Technology is getting more powerful, accessible, and capable every day, thereby increasing its attractiveness to the general public constantly.

Technological enthusiasts are avid about all these developments. They want to use and experiment with everything new and even see it is a challenge if something seems hard to use. The mere process of figuring out how to operate some new interface is exciting for this small minority.

However, the general public does not embrace new technologies, they just see them as a tool to do their work. If it does not work the way they think, they are getting frustrated and lose their interest quickly.

At the cutting edge of technology, too often systems are designed by experts for experts resulting in products like the early video-cassette recorders (VCR), which have not been usable by many, not even with training materials. In this special case a company saw and ceased an opportunity by marketing a third-party product which made recording "easy" by entering a several digit number, which could be taken from the TV guide, into a special remote.

The cost of these not very usable products can be astonishingly high. They result among other things in lost market share, low user productivity and high expenses in training and customer support.

Fortunately well-established techniques for achieving software usability are available by now, and companies started realizing their benefits by incorporating these techniques into standard product development methodologies [42].

However, this process usually only happens with comparatively "old" technology, once it has been firmly established in the market. Companies are pretty much forced to focus on improving the usability of their products, once the market has been saturated with many competing, similar products. As far as the user is concerned the product is the user interface

and therefore usability is an aspect immediately obvious to her. In fact it is a great way to distinguish one's own product line from rivaling ones.

Technologies like "Wearable Computing" have the vision of non-obtrusive computers everywhere by integrating them into clothes. Augmented Reality ([8], [44]) enhances user's senses with digital information augmented into reality. Ubiquitous Computing [75] aims at creating many small intelligent devices weaving them into the fabric of everyday life until they are indistinguishable from it (See Chapter 2).

When new technologies such as these are explored, aspects regarding their usability are often pushed into the background, resulting in products which are often not much more than technology demos. These usually generate interest in the technology itself, but disappoint once users try to seriously use them on a regular basis.

In my opinion, you cannot start thinking about the above mentioned problems concerning new technologies too early, so this paper will explore usability aspects for Ubiquitous Computing concentrating on Tangible Computing and Augmented Reality thereby shifting the focus away from the technology to usability concerns. It is an attempt to help create products based on these new technologies with good usability from the very start.

1.2 Scope of Thesis

The Problem

The mixture of the above mentioned new technologies poses new problems to usability engineering. Additional monitoring of the added context of the environment is for example required by Wearable Computing [38]. In Tangible Computing it has to be decided which objects should actually be made tangible, then combined with which digital representation, exerting which form of control in what level of constraints while defining which physical state [68].

The choice of interactions and metaphors is not trivial either [52]. Well defined methods are required to make a profound choice on the available options without merely relying on random choice, intuition or experience.

The nature of Augmented Reality implies that virtual objects are embedded into reality, which complicate proper monitoring by a usability engineer during usage, since these objects cannot be seen without special means. A suitable evaluation setup has to be found which makes monitoring in this environment feasible.

The ARCHIE[1] project, which this thesis is part of, has had the problem of usability nescience regarding the Chameleon Touch-Pad input device. There was no clear idea in which scenario this input device might fit nor how the interaction with it should take place maintaining good usability.

The underlying system DWARF[2], a framework for distributed systems, additionally makes data-logging harder than it would be in a monolithic case. Not only a method for

[1] **A**ugmented **R**eality **C**ollaborative **H**ome **I**mprovement **E**nvironment
[2] **D**istributed **W**earable **A**ugmented **R**eality **F**ramework

logging data in this system has to be established, it is also imperative to decide which data, from which meaningful results can be drawn, is to be logged.

Proposed Solution

The recurring task of usability engineering of Ubiquitous Computing should be simplified by the usability evaluation framework I propose in this thesis. This framework presents a step-by-step guide on how to develop new design guidelines by usability evaluation.

An implementation for data logging based on new services, which will ease future usability related data logging in DWARF in combination with new visualization tools, is detailed.

Recommendations for the improvement of ARCHIE usability were concluded from a sample usability study applied in the course of this thesis to verify the proposed evaluation framework.

1.3 Structure of Document

Part I "Foundation", the first of four major parts of this thesis, lays the groundwork by defining necessary terms to get started and by giving an overview about ARCHIE and its basis DWARF.

The following part II "Usability Engineering Process" concentrates on the process aspect, describing the usability evaluation process itself, after talking about process related requirements and work. The focus here is clearly on the process of usability evaluation, producing a usability evaluation framework, and not on the complete usability engineering life-cycle which is inspected only sketchily.

Similar to the previous, part III "Software Engineering" concentrates on the tool aspect, where the written software is presented in a software engineering type of way again after covering tool related requirements and work.

The final part IV "Sample Usability Evaluation" contains all details about an actual study, which was performed to verify the proposed usability evaluation framework.

A summary followed by future perspectives concludes the thesis.

2 Term Definitions

What are the most important terms? Actors are defined in this chapter.

This chapter defines terms frequently used in this thesis in detail. After covering new technologies and their relation, an overview is given about user interface generations. At last, usability engineering itself is defined which leads to the actor analysis.

The new technologies for human computer interfaces can be put into relation with Reicher's [48] design space for mobile systems as seen in figure 2.1.

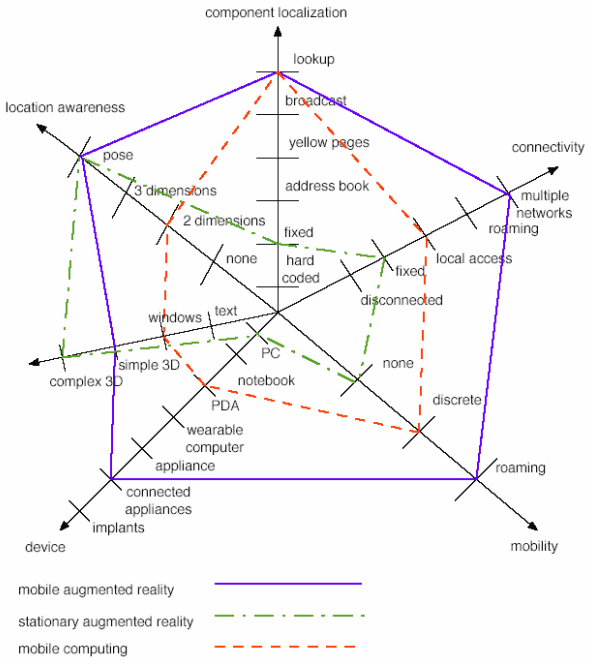


Figure 2.1: Design space for mobile systems (Courtesy of Reicher et al.)

The six descriptive dimensions user mobility, type of user device, network connectivity, component localization, location awareness, and media richness each describe one aspect with discrete values. The dimensions itself were derived from an analysis of existing Ubiquitous Computing, Wearable Computing, and Augmented Reality systems. Elements further

to the outer side are more complex, providing a simple schema to estimate the complexity of mobile systems.

Type of User Device This describes the device the user is working with. This goes from bulky devices, such as personal computers on to notebooks, personal digital assistants to wearable devices. Special appliances build for one specific purpose increase the complexity further. Cooperating appliances, copliances, are only topped by implants in the human body.

User Mobility How much can the user move with her device? None means the user is fixed stationary, discreet allows usage only at discreet points with no intermediate usage and roaming allows un-obtrusive usage everywhere.

Network Connectivity This dimension describes the quality and flexibility of the network connection, ranging from being disconnected, to fixed, very limited access. More complex variants are local access with hot spots for access and roaming where the user cannot be not online. Finally multiple networks describes the smooth passing in between different network types.

Component Localization Cooperating components need means to find each other. This can range from being hard wired together, to a fix address, or to asking a name server for the address. With yellow pages a server is queried for a component meeting certain criteria. Broadcasting just sends messages to every component in the network, allowing anyone to answer. Lookup, the most elaborate technique for localization combines the yellow pages with the broadcasting method to find the yellow page server.

Location Awareness This dimension describes the level of concern of the application regarding the user location. This can range from none, to flat two-dimensional, to three-dimensional also taking the altitude into account. Pose fully specifies the user's position and orientation required for Augmented Reality applications.

Richness Of User Interaction This dimension explores the media richness, which can be simple text, GUI-like window based, simple 3D for augmenting the user's vision with virtual objects with no attempt to fool reality. Complex 3D aims at augmenting reality perfectly so the user won't be able to distinguish between real and virtual anymore.

2.1 Wearable Computing

The idea of the least complex form "Wearable Computing" is that the computer is worn by the user like a piece of clothing offering him access to information without the need to leave his task. This is achieved by small form factors combined with advanced input and output devices.

See figure 3.8 on page 32 for an example on Wearable Computing.

Increasing the complexity to the next paradigm we get to Augmented Reality.

2.2 Augmented Reality

Like mentioned earlier, this thesis is part of a project to develop an Augmented Reality application.

For many people the familiarity with Virtual Reality (VR), where a user is completely enclosed in a purely virtual scene, e.g. for flight simulation, is usually higher than with Augmented Reality. Due to their nature, these systems take place in some sort of CAVE (Computerized Automatic Virtual Environment), thereby restricting the movement range of their user considerably.

Augmented Reality systems on the other hand merely complement or *augment* the users' reality by adding virtual objects to it. Reality which is augmented in this sense conveys additional information, in effect enhancing the users' senses to see e.g. a future building at the real, still empty construction site, whereas VR would only be able to show the future building but not in the context of reality. Different forms of augmentation are possible, like visual, audio and tangible.

A survey about Augmented Reality application by Azuma [8] concluded in the following definition:

> *AR are systems that have the following three characteristics:*
>
> 1. *Combines real and virtual*
> 2. *Interactive in real time*
> 3. *Registered in 3–D*

Milgram has offered a more concrete definition in form of the "virtuality continuum" (Figure 2.2, [44]) which connects completely real environments to completely virtual ones. Mixed Reality (MR), which includes Augmented Reality refers to everything in between. Moving from left to right the amount of virtual imagery increases and the connection with reality weakens. "Augmented Virtuality" is in reference to completely graphic display environments, either completely immersive, partially immersive, or otherwise, to which some amount of (video or texture mapped) "reality" has been added.

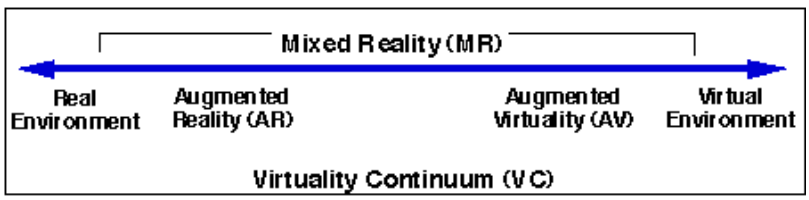


Figure 2.2: Virtuality continuum (Courtesy of Milgram et al.)

An example visual augmentation is shown in figure 2.3.

Figure 2.3: An Example of Augmented Reality

2.3 Ubiquitous Computing

Ubiquitous Computing takes the complexity up one notch.

Building on a new way of thinking about computers in the world, one that takes into account the natural human environment, Ubiquitous Computing aims at creating a world in which people interact with and use computers without thinking about them, by weaving them into the fabric of everyday life until they are indistinguishable from it.

This vision of Mark Weiser replaced the one-size fits all personal computer model with a variety of task specific devices at different locations of different scales. The term "embodied virtuality" is used, to refer to the process of drawing computers out of their electronic shells to achieve true ubiquitous, invisible computing.

The real power of the concept comes not from any one of these devices; it emerges from the interaction of all of them [75].

Examples for visionary Ubiquitous Computing can be seen in the following figure 2.4.

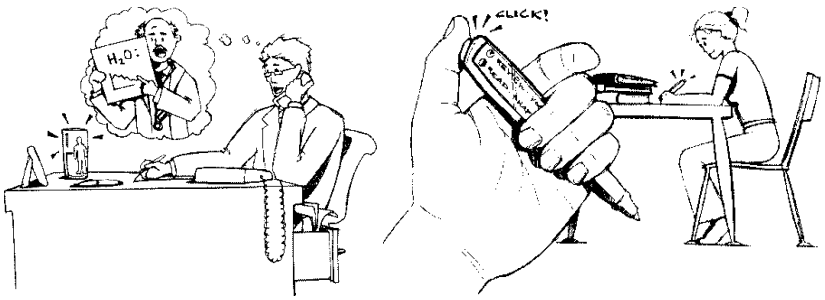


Figure 2.4: Examples for Ubiquitous Computing(Courtesy of Fogg et al.); Left: Study Buddy, Right: HydroTech

The “Study Buddy” might look like a normal pen but it is designed to influence students study habits. The device remembers learning sessions and makes intelligent suggestions, keeping the student at track of work. It might also be used to show e.g. pulsating shapes of currently learning colleagues or vibrate when a special study colleague enters the library to motivate to study too.

The HydroTech is the vision of a medical device helping people not to forget drinking when it’s important for their health. A transmitter to measure dehydration is implanted in the patient sending data to the medical faculty for monitoring and to the special water glass which shows the current level of dehydration via a color changing body image [12].

2.4 Generations of User Interfaces

As mentioned before the user interfaces of human computer interaction have changed radically over time. The first systems have been very primitive allowing minimal interaction, whereas the newer interfaces allow more and more interaction in a increasingly intuitive fashion.

Beginning from the oldest user interface a short overview is given.

Batch Systems

The term originated with mainframe computers when punched cards were the usual form of computer input. The computer operator placed a batch of cards (one batch per program) in a box, in the sequence that they were to be fed into the computer. There has been no option of interaction during the whole processing phase. If any error occurred, the whole process had to be restarted.

Still today there are batch jobs, which are programs assigned to run without further user interaction such as printing jobs or system maintenance operations usually scheduled to run in the night.

Command Line Interfaces

Allowing more interaction than batch systems, command line interfaces give the user the option of entering lines consisting of commands, which are to be remembered. There is no feedback from the system, until the line has been confirmed. If there was an error in the line, the line has to be repeated with the corrected command line syntax. Typical command line interfaces can be seen in figure 2.5.

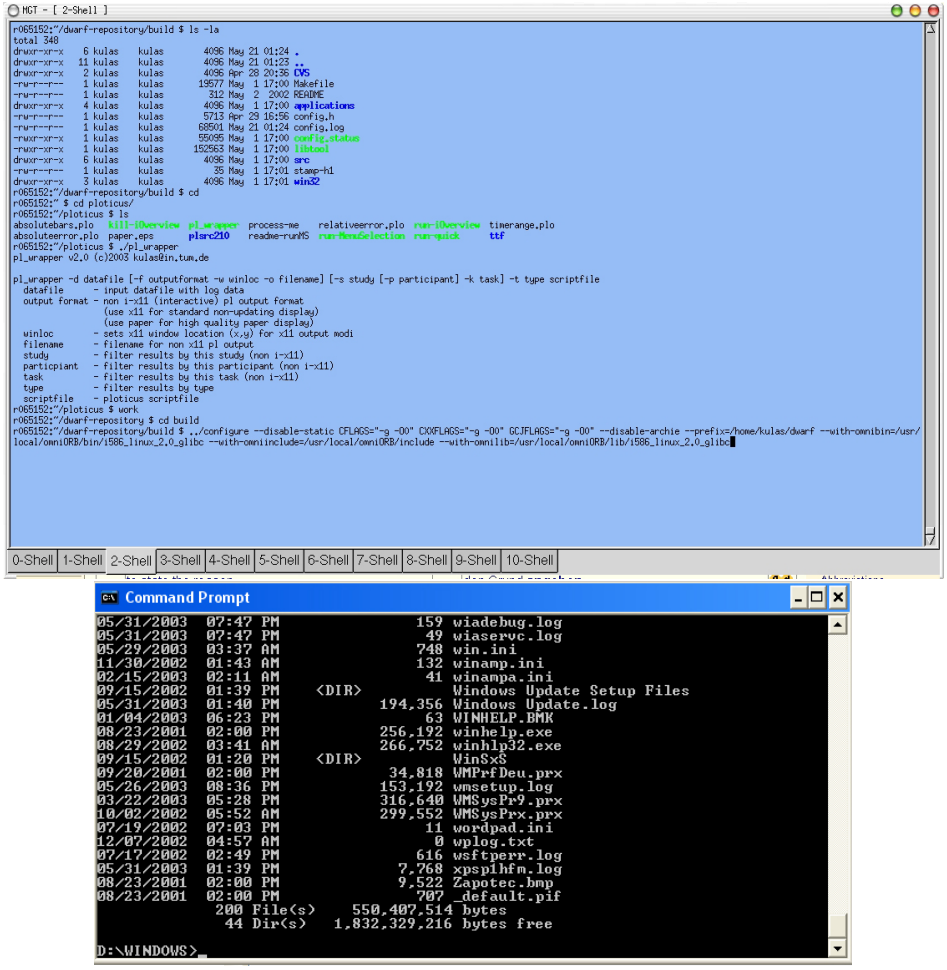


Figure 2.5: Typical command line interfaces, Top: Linux; Bottom: DOS

Although usability engineering is already reasonable in command line interfaces, it starts to get much more interesting beginning with the next iteration of user interfaces.

Graphical User Interfaces

Skipping non-graphical menu-based interfaces, which just added a mouse, graphical user interfaces (GUI) are today the most common human computer interface.

These interfaces rely on some form of mouse and keyboard input, which are used to manipulate typical widgets such as buttons, sliders, radio-buttons, and pull-down menus altogether usually arranged in windows. This is why they are also often referred to as WIMP (windows, icons, mouse, and pull-down menu) interfaces [60].

The term “Graphical User Interfaces” was originally coined by Apple with their first graphical user interface in 1983. These systems have been copied and refined ever since but essentially they have stayed true to their original nature for the last two decades.

Such interfaces look typically similar to those in figure 2.6.

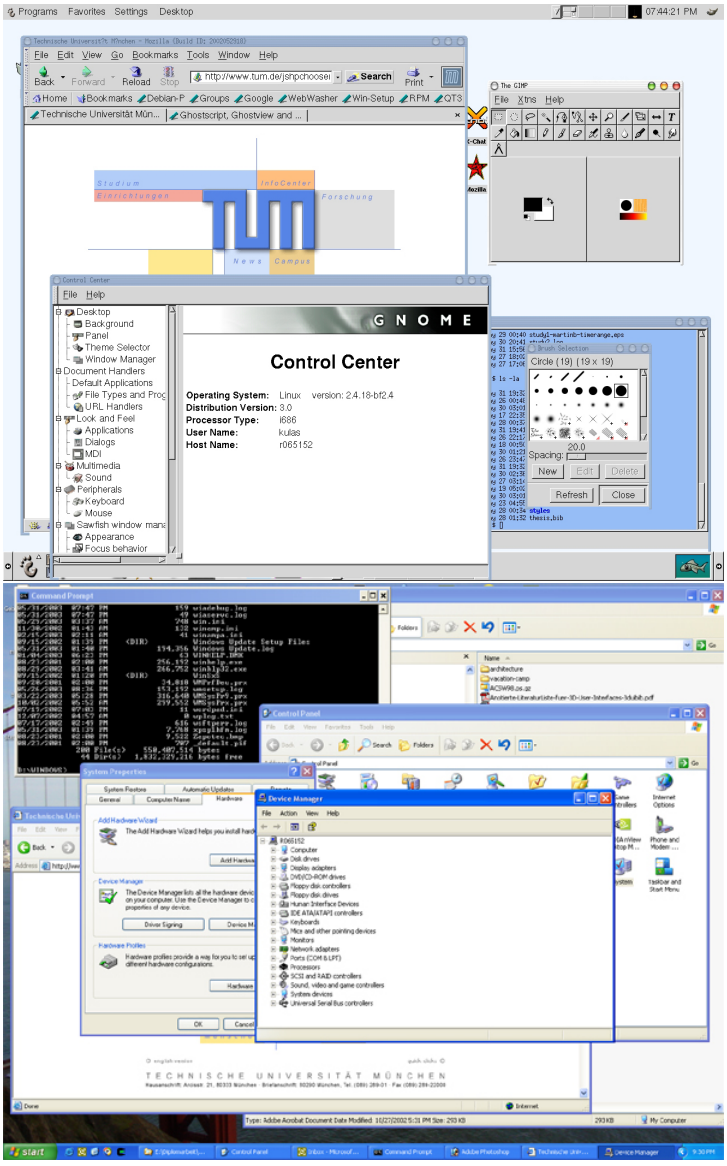


Figure 2.6: Typical graphical user interfaces, Top: Linux; Bottom: Windows

Tangible User Interfaces

Next-generation interfaces like tangible user interfaces take a very different approach to traditional graphical user interfaces.

Ishii coined the term "tangible user interfaces" with his MCRpd TUI interaction model (Figure 2.7, [68]).

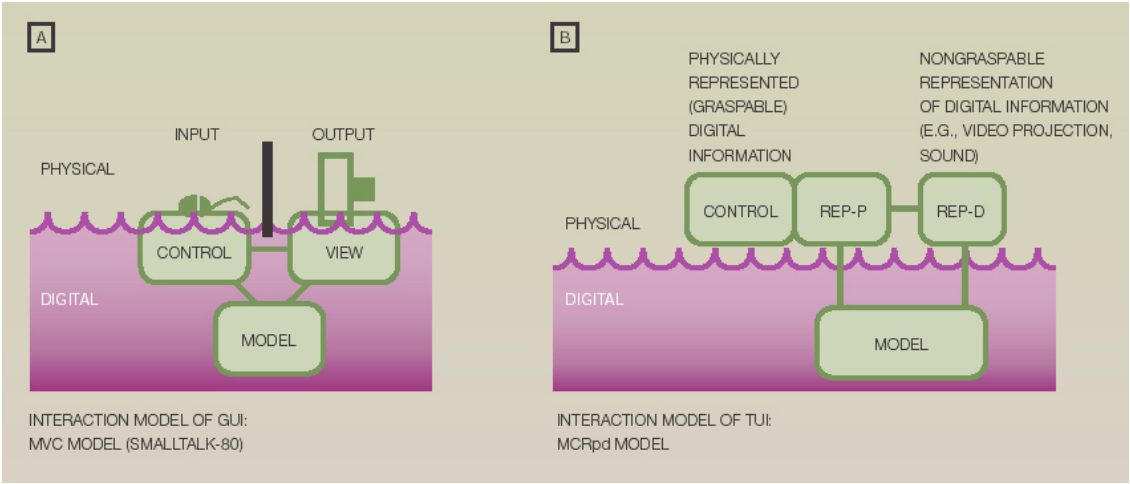


Figure 2.7: Tangible MCRpd Model (Courtesy of Ishii et al.)

Traditional graphical user interfaces share a clear distinction between the visual representation (or *view*) provided by the graphical display and the *control* capacity mediated by the mouse and keyboard of the UI.

This border between view and control blurs for tangible user interfaces as expressed in the model-control-representation (physical and digital), or MCRpd. The *view* notion has here been replaced with the notion of a physical representation (*rep-p*) and a digital one (*rep-d*) highlighting the TUI's integration of physical representation and control.

A sample for these user interfaces is "Illuminating Clay", a system for real-time computational analysis of landscape models as seen in figure 2.8. Here users alter the topography of a clay landscape with their hands which changes the projected result of an arbitrary landscape analysis function in real-time [46].

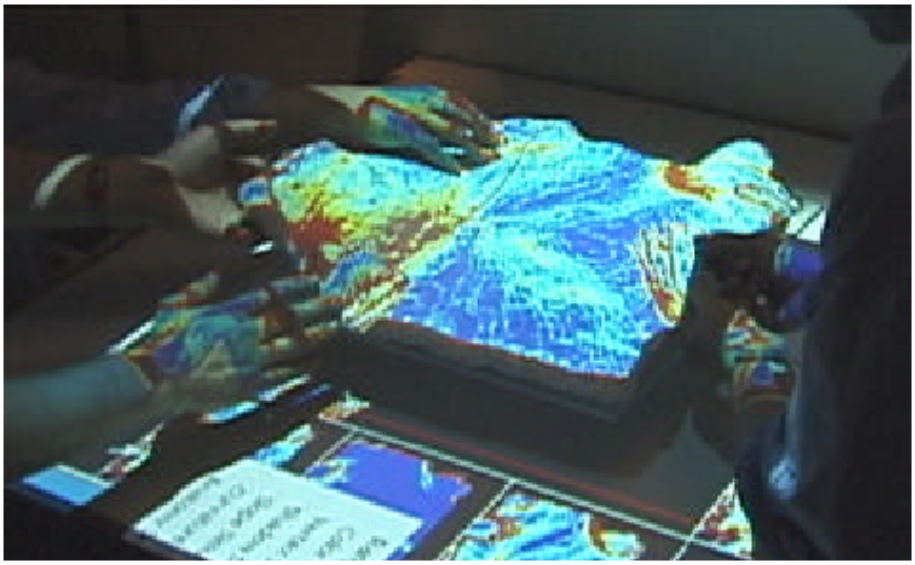


Figure 2.8: Illuminating clay: User's hand interacting with physical and digital representation (Courtesy of Ishii et al.)

2.5 Input Device Taxonomy

For the later chapters it is helpful to understand the input device taxonomy which was developed for ARCHIE [76] as seen in figure 2.9.

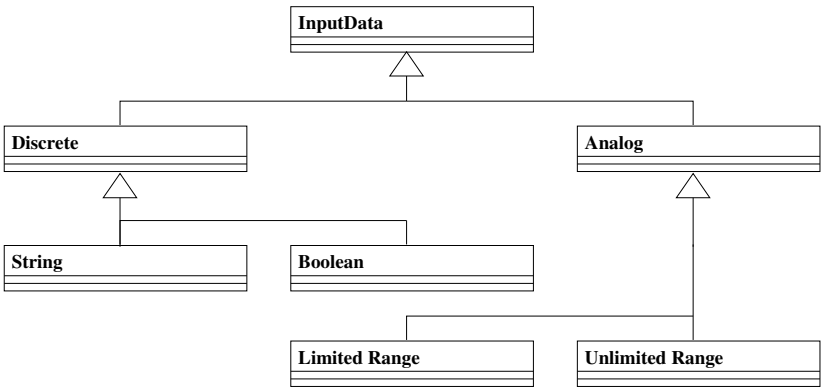


Figure 2.9: Input Device Taxonomy

- Boolean: All discrete data can be decomposed into booleans. Buttons, for example, are mapped to boolean.

- Limited Range: Every analog value with a lower and upper bound can be mapped to a Limited Range entity. This entity has always a range between 0 and 1. Examples are sliders or angles.
- Unlimited Range: Every irrational number which has only one or no bound. Examples for this are coordinates.
- String: Text strings.

2.6 Usability Engineering

What is usability engineering about?

As defined by Mayhew [42]:

> *Usability Engineering is a discipline that provides structured methods for achieving usability in user interfaces design during product development. It is a discipline with roots in several other basic disciplines, including cognitive psychology, experimental psychology, ethnography, and software engineering.*

Exploiting the knowledge about human information processing and using empirical methods to measure and study human behavior, products are designed in a way so they are easier to learn, easier to use, and otherwise tailored to optimally support the specific target users doing the specific work tasks a product is intended to support.

Actors

Multiple actors are involved in the process of usability engineering [42]. We will now identify the most important actors which matter for this thesis.

Actors represent external entities that interact with the system. An actor can be human or an external system [15].

The development of the UIML language [3] aims at adding a layer for user interface development, analogous to traditional APIs. This allows designers to describe user interfaces in a highly appliance-independent manner, shielding them from implementation details. Another big advantage is the possibility to develop different user interfaces for the same program with relative ease, thereby offering different views on the same model of data.

That means programmers implement the internal logic, while experts on user interface design such as human factor specialists, graphic artists or cognitive psychologists can focus on the user interface itself.

Since we focus on design questions in this thesis programmers concerned about implementation details are not to be taken into our list of actors.

We identify the following actors who come into contact with the proposed framework:

Evaluation Monitor This person combines all the roles of conducting usability studies in one. He selects, briefs, and debriefs the participant for the study, prepares the evaluation materials, conducts and logs the actual study and finally analyzes and evaluates the results. He is exposed to the usability evaluation framework to a great deal.

Participant Representative of the target population (end-user) who uses the system while being observed by the evaluation monitor.

Design Team This team is responsible for deciding on the user interface design of a given product. They include all the above mentioned experts.

Management This person is barely involved in the framework, but receives the analyzed results from the evaluation monitor which have been prepared with the framework and reaches new derived decisions.

It is not surprising that the above listing only contains actual persons, considering the area of usability engineering where they were taken from.

3 DWARF and ARCHIE

The group project ARCHIE and its basis DWARF are introduced.

This chapter provides a general overview about Augmented Reality projects and frameworks, in particular DWARF the Augmented Reality approach of Technische Universität München. After over viewing the guidelines that lead to the DWARF framework its current state of development is outlined.
At the end of this chapter the ARCHIE project is introduced as a group project of several SEPs[1] and diploma theses. The completion of the ARCHIE project provides new functionality to DWARF thereby making it more mature.

3.1 Augmented Reality: A Stake-holders Point of View

The original intent in the development of computers was to provide support to people whose work was too difficult or too lengthy to solve manually, like large mathematical equations. New technologies arose as computers gained speed and more peripherals were connected to them. But the basic intention kept the same. Computers are supportive tools.
The increasing spread of computer networks in the last decade of the 20th century allows the distribution of services allocated to specific tasks. For example, rendering of 3D scenes is a resource intensive procedure which can be separated to another hardware, while a second machine can handle necessary remaining tasks of an application. The distribution of dedicated services to various platforms can get used in the Augmented Reality domain, because applications using this discipline have to aggregate various areas of computer science, where each may require a lot of computation.

Using Augmented Reality to support people can happen in diverse ways. But for the discipline of Augmented Reality two classes of computational assistances can be identified. On the one hand, there are independent tasks that can be supported by Augmented Reality, while on the other hand, the diversion of computers through the environment provides resources for Ubiquitous Computing. Both classes are described and in advance a combination is described.

[1] System Entwicklungs Projekt - a project every computer science student at TUM has to absolve

3.1.1 Independent Tasks

Closely focused on a task, users may perform task-centered activities like maintenance or navigation [10]. To realize applications of this kind, developers can rely on paper based guidelines like maintenance guides or city maps. These guides can get formalized in state machines executed by taskflow engines [54]. And the Augmented Reality application leads the user through the task step by step.
Because of the runtime environment being known in advance, applications are comparatively easy to realize. Due to their nature these applications provide no flexibility to the users. Only the specified task can be realized usually only in the location specific to the application.

3.1.2 Intelligent Environments

The combination of task-centered Augmented Reality and Ubiquitous Computing can result in Augmented Reality-ready intelligent environments. The aggregation of both aspects supplies services provided by the environment. Seamless interaction between these services and a mobile Augmented Reality system give each user a way to dynamically perform tasks as described in section 3.1.1.
For example, as the user enters or leaves a room, his Augmented Reality system recognizes context changes and informs the user about new services. Options could be offered via aural or visual channels. A head mounted display (HMD) can display information of tasks available from the current location. Also the HMD can be utilized to visualize the chosen applications' user interface by rendering e.g. virtual 3D scenes. If a corresponding tracking service is available the user can leverage this to get an accurately aligned view matching the current perspective.
Systems using such intelligent Augmented Reality-enabled environments are powerful, as they can accommodate not only predetermined taskflows but also spontaneous desires by the user.

3.2 Related Work

At the current time there are several research projects on Augmented Reality all over the world. The resulting software architecture of the systems differ wildly ([69], [70]), but two general directions can still be seen.

In the first one prototypes are built by research groups which often result in task-centered systems for e.g. basic tracking concepts or car development concepts [72]. Usually they are highly specialized and monolithic. Many of these systems provide small demonstration setups for particular tasks. Even though the realized tasks essentially have a similar focus in other systems, the re-usability of their technology is quite difficult.

Other projects focus on middleware technology covering central Augmented Reality tasks and by this provide frameworks for applications. Although the concepts of software engineering [15] have been known for some time, they have not been widely applied in the Augmented Reality problem domain. But there are some projects tackling this issue.

The Computer Graphics and User Interface Lab of Columbia University has assembled different types of Augmented Reality applications [20] from a common basis. Their work focuses on providing a common distributed graphics library as a framework for reusable software components.

Mixed Reality (MR) Systems Laboratory of Canon Inc. developed a basic frame for mixed reality applications [72]. Their set includes HMDs and a software development toolkit for building MR/AR applications. The provided library supports common functions required in Augmented Reality applications, but still it cannot be spoken of a framework.

German Ministry of Education and Research founded the project ARVIKA[2] which is primarily designed as a Augmented Reality system for mobile use in industrial applications. The architecture is user centered, but relies on fixed workflows. It does provide a configurable access to offered features.
The industry, in particular a sub department of the Volkswagen AG needing software engineering technologies, already used some basic ARVIKA systems for car crash simulations [72].

An example for a multidisciplinary research program is *UbiCom* (Ubiquitous Communications) from the Delft University of Technology. Their architecture combines mobile units with stationary computing servers and focuses on mobile multimedia communications and specialized mobile systems [36].

Another approach is lead by the *Studierstube* project at Vienna University of Technology. That group uses concepts of software architecture among other things, but only as far as to keep parts reusable for testing new user interface paradigms [59] or for reconfiguring the tracking subsystems [50].
This can however only partially be seen as a framework for multi-user and multiple applications in Augmented Reality.

At last we will take a final view on projects about Ubiquitous Computing. Today several approaches and technology systems share the idea of providing services to users through a star-shaped architecture, such as Ninja [27] or GaiaOS [29], [55].
Extendible Augmented Reality frameworks should rely on a decentralized architecture instead of the architecture of the approaches of these projects. Although some of them providing service federation, they don't seem to offer context and configuration propagation.

[2] www.arvika.de

3.3 DWARF

The Technische Universität München also has a research project on Augmented Reality, which is called DWARF. The name is an acronym representing the guidelines for the general system architecture. DWARF stands for **D**istributed **W**earable **A**ugmented **R**eality **F**ramework.

The DWARF infrastructure provides an extensible, flexible and modular software framework for reusable Augmented Reality relevant components.
The framework can be seen in four abstraction levels. Figure 3.1 shows that layers.
The bottom layer is the layer of dynamic peer to peer systems. It provides connectivity and communication mechanisms for processes.
On top of this layer, the solution domain resides, supplying general components for the domains of Augmented Reality, wearable and ubiquitous computing. Services for tracking and environmental context are located here.
The third layer is described by the application domain. Components reusing general tasks of the sublayer reside here.
The top layer is built by the concrete applications available for the users.

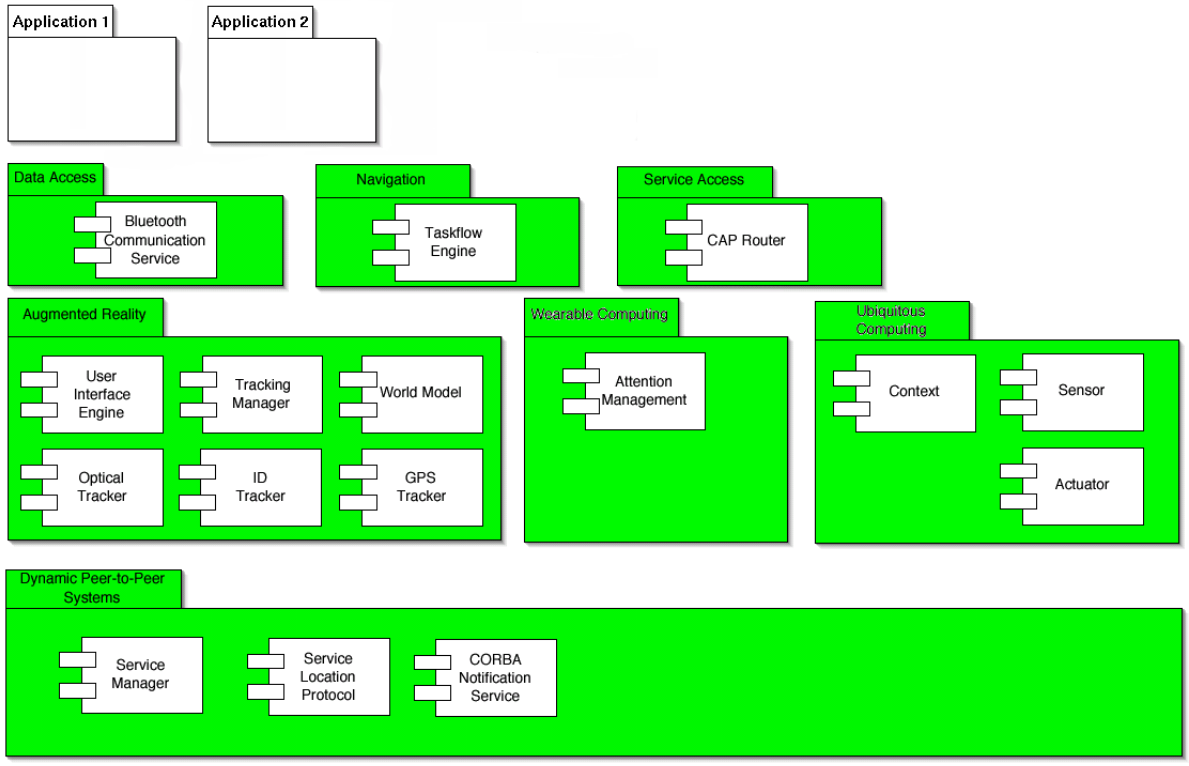


Figure 3.1: A layered architecture for the DWARF framework

A suitable framework for the Augmented Reality domain has three aspects: the announced *services*, a connecting middleware and a common architecture providing a basic

concept to enable applications [10]. This framework allows components to be reused between applications and to dynamically configure them. One could e.g. imagine that the same tracking system provides position and orientation of certain objects to different applications.

Services *Services* are dedicated components covering general Augmented Reality tasks. Each *service* provides certain abilities to the user or to other *services*. On the other hand they can rely on input from other *services*, suppling filtered, analyzed, and rebuild information to other components or users.

Middleware A distributed application dynamically matches, connects and configures *services*, so that these can communicate directly corresponding to their needs respectively their abilities. The amount of *service* and their combinations can be changed dynamically by the middleware at runtime if required.

Architecture A conceptual architecture describes the basic structure of Augmented Reality systems that can be built with it. To properly integrate different *services* with each other the respective developers have to agree on the roles of their *services* and on interfaces between them.

For our realization at TUM the main part of the framework consists of the *service-manager* and the service communication infrastructure. DWARF is designed as a distributed, and thereby wearable framework. Personalized software components can reside on different hardware components, even on mobile devices [10], enabling Ubiquitous Computing [75]. The *service-manager* handles all *services* and dynamically starts and stops required components on the corresponding hardware platforms.

The following section describes this context particularly.

3.3.1 Services

At DWARF applications each *service* can potentially run on its own hardware device as an independent process. It is described in a *service description*. Additional information about service parameters is stored here using attributes. There could, for example be an attribute `accuracy` for a tracking device, or a `location` attribute for a printer.

In DWARF those *service descriptions* are specified in conjunction with *needs* and *abilities*. These describe on a high level how *services* can connect. Looking at two connected *services* one has an *ability* and the corresponding partner has the matching *need*.

Each of these two has a *connector* specifying the communications protocol between them. Current protocols provide communications via event notifications, remote method calls

(CORBA[3]) and shared memory. Two communicating *services* must have the matching communication protocols.

Needs and *abilities* also have a `type` which distinctly defines the corresponding interface. Thus for two matching *services*, one has a *need* with the same *connector* and the same `type` as the other services' *ability* has.
Hence `types` in *needs* and *abilities* can be used in various ways, e.g. a selection predicate is settable. The coding rules for these predicates follow the LDAP RFC 1558, 1960, 2054[2].

By the use of `minInstances` and `maxInstances` values for multiplicity are attributable to *needs*. If for example `minInstances` is set to "1" for a *need* of a *service*, this *service* will only start properly when at least one corresponding ability of any other *service* is connected to it.

Figure 3.2 illustrates the description of two different *services* with a possible connection in easy readable XML-notation.

```
<service name="Tracker"
         startCommand="Tracker"
         startOnDemand="true" stopOnNoUse="true">
  <attribute name="location" value="GreatHall"/>
  <ability name="peoplesPositions" type="PoseData">
    <attribute name="accuracy" value="0.1"/>
    <connector protocol="PushSupplier"/>
  </ability>
</service>

<service name="Map">
  <need type="PoseData"
        predicate="(&amp;(location=GreatHall)(accuracy<1.0))"
        minInstances="1" maxInstances="10">
    <connector protocol="PushConsumer"/>
  </need>
</service>
```

Figure 3.2: Two simple connectable service descriptions

3.3.2 Middleware

A *service manager* residing in each participating computer, is able to contribute its service descriptions to a common pool. The service-managers internally check all possible connections between *needs* and *abilities* of all *services* and dynamically connect and start matching ones on demand. [39]
Intra-*service* as well as internal *service-manager* communication take place via CORBA. Thus every *service* contains an interface to it.
The *service-managers* running on different computers find each other via SLP[4].

[3] Common Object Request Broker Architecture
[4] Service Location Protocol

3.3.3 Architecture

A conceptual architecture defines the basic structure of Augmented Reality systems which can be constructed with it.
Thus it ensures that service developers agree on the roles of their own *services* within the system and on interfaces between them.

Figure 3.3 shows an example architecture for DWARF applications. It is separated into six packages. The distribution of services among the required subsystems of the general Augmented Reality architecture is shown, too.
The tracking subsystem is responsible for providing location information on real objects as positions or raw data streams. The world model subsystem holds all relevant data on real and virtual objects and provides virtual models for the presentation subsystem which generates user visible scenes. Interaction with the system is handled in the control subsystem that reacts on user input and provides input data to the current active applications. The application workflow resides in the application subsystem. Context information is handled by the context subsystem.

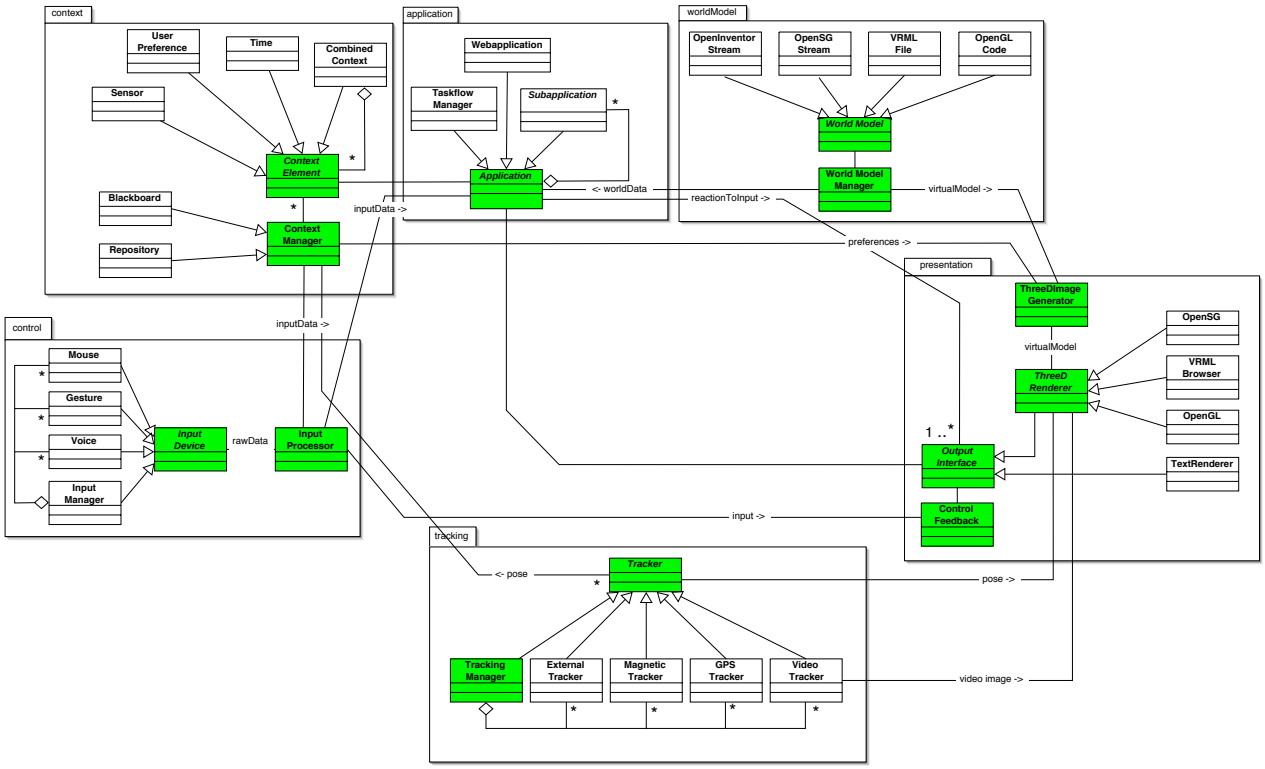


Figure 3.3: General DWARF architecture

3.4 Extending the Space of Components

The DWARF framework is still under development. However multiple applications have been built by now with it.
Its initial version was verified by implementing an application for a guidance scenario called *Pathfinder* ([9, 39, 43, 54, 58, 73, 77]).
Following this, multiple projects (*STARS*,*Fata Morgana*, *FIXIT*, *TRAMP*, *PAARTI* and *SHEEP* [1]) increased the amount of services, provided by the framework.

3.4.1 Existing Services

Besides the classification to different levels in the last section, services can also be classified in four groups. One will see that some groups directly reference reusable framework components, others facilitate development while a group exists due to the development process of the framework. That group arose because the interests in the previous projects remained on the development of other components and demonstrative setups for applications, these components were use in.

Application Specific and Conceptual During the initial development stage of a new framework, not all requirements can be met. Thus to write a full-blown application some static services might be required to make up for missing framework functionality. With a perfect, stable, and feature complete framework, services like that will be obsolete.
Conceptual services demonstrate basic functionality and allow developers new to the framework to quickly familiarize themselves with it by looking at the code.

Testing A testing service assists the developer in debugging other services by simulating not yet fully implemented partner services by providing fake data. A tracking service could for example be tested with a black box test [15], made up by a test service which simply receives and displays the tracking data.

Task-focused Task-focused services are usually low-level services. They provide information on dedicated tasks. They can be configured in the range of their activity. Trackers for instance, provide position/orientation information on certain objects. Which objects to track can be specified, but the tracker always will track objects and provide their location information.
In DWARFs current state an ART[5] system is such a task-focused service as well as for example a sound-player service.

Generic Components configurable to handle various interfaces and data-types. In contrast to task-focused components, these services provide a specific basic functionality on workflow activities that is configurable to required tasks. Their interfaces are adjustable to

[5] Advanced Real-time Tracking - a commercial optical tracking system

receive and provide task required data-types.
In this group the framework provides the User Interface Controller.

A goal of the development of a reusable framework for mobile Augmented Reality applications is to decrease the amount of components classifiable to the application specific area and to increase the amount in the other classes supplying configurable services.
Actual and new applications are made up by a combination of a set of configured generic services which require task-focused services during runtime. Current existing applications also rely on application specific components and new ones will, too. But their amount is intended to decrease in future.

3.4.2 A Requirements Generating Project

Ongoing development in university projects produced a usable framework with various components. But the DWARF framework is still far from being complete.

New properly calibrated input and output devices were needed. Also model for configuration and data persistence was required. Systems acting in intelligent environments also need components enabling the user to select services and providing Augmented Reality views on 3D-objects. The mobility aspect shall receive a contribution in form of a user-attached context aware service.
Finally, as research projects often result in unusable systems, human-computer interaction components should be evaluated for usability. So that the customization of these components to unskilled users takes not a great burden to them.
The infrastructure also required major improvements, to provide new generic middleware features like multiple access via shared memory, template-services, and dynamic configuration bypassing[6].

These technologies extend the framework making it more suitable for future projects. During development of the mentioned project, new questions arose which make further improvements on the components possible.

While developing reusable framework components for a distributed dynamic system, the requirements for possible real applications should be kept in mind. The resulting architecture should be flexible enough to not only allow the development of the proposed new application but also support the development of completely different DWARF applications. Both approaches keep up continuous requirements engineering.
During the applications development process in the concrete project the requirements for the framework components get validated.

[6]Though a documenting paper about DWARF middleware is being written, these features are documented on the developers board only at the time of this writing. Additionally the feature implementation is still in an experimental phase

A new team project was planned in July 2002 to provide a number of missing components. The aggregation of multiple student study papers into team projects proved to be very rewarding for all participating members in the history of DWARF. Members learned about team-work, building large systems and could practically experience their skills. Also the production process is more interesting than in a solo project, because a system is built, on which others rely and that will be reused by other projects.
Keeping this in mind, the ARCHIE project was founded in a workshop in Konstein in the summer of 2002. In combination with the current requirements for the framework, suitable scenarios were found to give application specific requirements for the production of the new components.

3.5 ARCHIE

This section introduces the ARCHIE project. The name is an acronym for **A**ugmented **R**eality **C**ollaborative **H**ome **I**mprovement **E**nvironment.

ARCHIE is an interdisciplinary project between the Universität Stuttgart represented by Manja Kurzak, a student graduating in architecture [35] and a team consisting of seven members from the Technische Universität München. Manja Kurzak provided information about design processes and the planning of the construction of e.g. public or private buildings. Her information is summarized in the following problem statement section.

This project provided a starting point for requirements elicitation on the different single projects of all team members. It was intended to use ARCHIE in a prototypical implementation to give a proof of the requested components and their underlying concepts. To build an application with full functionality for architectural requirements was never a project goal.

The realized concepts have been shown to stakeholders like real-world architects in a live demonstration of multiple scenarios. In new areas like Augmented Reality new requirements are often generated by the client when the capabilities of the technology get apparent. To prove the flexibility and extendibility of the new DWARF components the chosen scenarios are partly independent.

3.5.1 Problem Statement

A *Problem statement* is a brief description of the problem the resulting system should address[15].

Usually a number of people with different interests are involved in the development process of a building.
The potential buyer has to mandate an architectural office to initiate the building process because the process is too complex to handle for himself. A mediator is responsible to

represent the interest of the later building owners towards the architect. The architect assigns work to specialized persons such as for example, technical engineers for designing plans of the wiring system.
Although the later building owner is the contact person for the architects office, he is only one of the many stakeholders interested in the building. Furthermore landscape architects have to approve the geographic placement of the new building in the landscape. Last but not least a building company has to be involved as well.

The architectural process itself is divided into multiple activities which are usually handled in an incremental and iterative way, as some specialized work goes to extra technical engineers, who propose solutions, but again have to reflect with the architects office.
After taking steps of finding and approximating the outer form of the building fitting in it's later environment, the rudimentary form is enhanced to a concrete model by adding inner walls, stairs and minor parts. This is followed by adding supportive elements to the model like water- and energy-connection, air-conditions, etc. As mentioned, this work always has to be reflected to the architect, because problems might occur during the integration of the different building components. For example the layout of pipes might interfere with the layout of lighting fixtures or other wiring. Spatial representation enhances pointing out problematic situations.
When the plans are nearly finished, the builder needs the possibility to evaluate the plan feasibility and if in the end all issues are resolved, all necessary plans can be generated and the builder can start his work.
In addition to that the building owner always needs view access to the model during the design phase. He wants to see his building in its environment. End users should be given the option to give feedback during the design phase too. So, the architects office receives feedback from many participants about their plans.

There are some entry points for Augmented Reality. The ARCHIE project delivers proof of concept for these aspects.
The benefits of the old style architectural design with paper, scissors and glue allows direct spatial impressions, while modern computer modeling does not provide these feature. Augmented Reality can bring these back to computer modeling.
Since preliminary, cardboard box models can not get scaled to real size and virtual models reside on a fix screen, public evaluations are difficult to handle. Via abstraction of position and orientation independent sliders could be used to modify views. But 3D steering of virtual viewpoints is not as intuitive as just turning a viewers head. Adding tangible objects as cameras provide familiar access to evaluation features.
User interactions with modern architectural tools require practice. So intuitive input devices would be useful.
Also in place inspection of building plans and models would be a great benefit for all participating persons.

3.5.2 Related Work

This section lists research projects of other groups working in the Augmented Reality as well as in collaborative systems domain. Although focus remains on architectural tasks, the introduced projects aim on consulting our team in ideas and concepts transformable to the ARCHIE project.

The international project *Spacedesign* [21] resulted in a comprehensive approach using task-specific configurations to support design workflows from concepts to mock-up evaluation and review. The implementation allows free form curves and surfaces. Visualization is done by semi-transparent see through glasses which augment the 3D-scene. The approach allows cooperation and collaboration of different experts.
This project focuses on a stationary setup, useful on the task of car development, while ARCHIE should also be usable as a mobile setup.

A single system combining mobile computing and collaborative work [51] has been built by *Studierstube* [59]. The system, assembled from off-the-shelf components allows users to experience a shared space, while there are still synchronization requirements for late-joining partners. The result is far from a reusable context aware framework since it does not provide any kind of location awareness.

The *Collaborative Design System* [7] developed by Tuceryan et al. at the ECRC[7] demonstrates the interactive collaboration of several interior designer. The system combines the use of a heterogeneous database system of graphical models, an Augmented Reality system, and the distribution of 3D graphics events over a computer network. As shows in figure 3.4 users can consult with colleagues at remote sites who are running the same system.

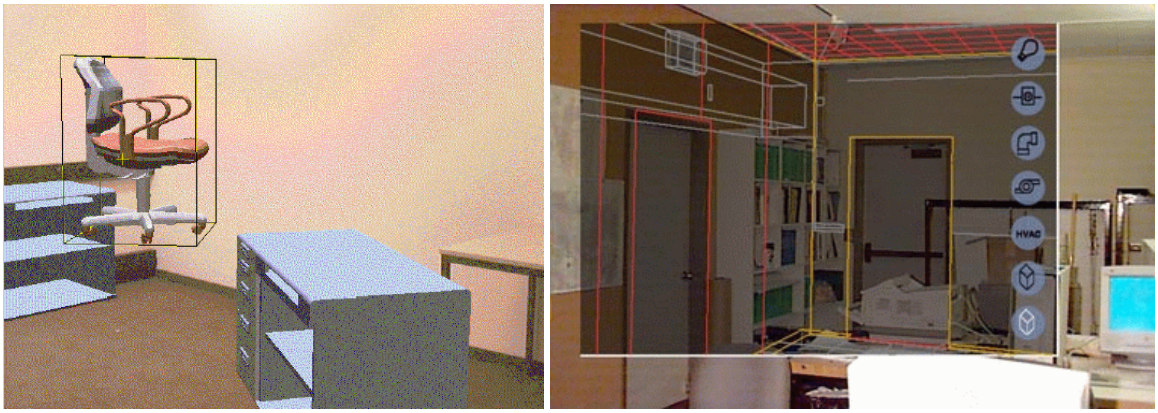


Figure 3.4: Left: *Collaborative Interior Design* [7] local user lifts chair while a remote user moves the desk; right: *An Application for Architecture* [67] overlapping a real office room with a CAD model

[7] European Computer-Industry Research Center

An architectural application [67] developed by Tripathy allows the user to view an architectural model within an Augmented Reality environment. The object model can be imported from any CAD[8] application, but it can not be modified. Though several additional information of real world objects can be seen, like the wiring of the lighting, or maintenance instructions for changing the bulb. Within the maintenance record the user even can see when the bulb was last replaced.

Webster et al. developed an application for construction, inspection, and renovation. The idea of supporting architectural tasks with Augmented Reality is not new but has already spawned testbed projects such as "Architectural Anatomy" [74]. Architectural Anatomy leverages Augmented Reality by giving the user a x-ray vision which might e.g. enable maintenance workers to avoid hidden features such as buried infrastructure, electrical wiring, and structural elements as they make changes to buildings and outdoor environments.
The prototype application tracks the user with an ultrasonic tracking system and uses a head-mounted display for monocular augmented graphics. The project aims at building systems that improve both the efficiency and the quality of building construction, maintenance, and renovation.

3.5.3 Scenarios

A *Scenario* is a concrete, focused, informal description of a single feature of a system. It is seen from the viewpoint of a single user [15].
This section describes the Augmented Reality relevant scenarios of the ARCHIE system. The following list does not describe a full architectural system, because the ARCHIE project is only intended to be a baseline for the development of reconfigurable DWARF services.

Scenario: **Selecting Current Task**

Actor instances: `Alice:User`

Flow of Events:

1. Alice enters her laboratory which contains the necessary environment for the ARCHIE application such as a *ARTtrack 1* tracking system and a running *service manager* on at least one computer. Furthermore she is wearing a backpack with several objects mounted on it. There is for example a laptop that provides the viewing service for her HMD. She also holds an *iPaq* in her hand.
2. As her *iPaq* attains the range of the wireless ARCHIE-LAN, its *service manager* connects to the one running in the laboratory, and they exchange their service information to each other. Now the *selector service* knows the available applications, and the menu pictured in figure (3.5) is displayed on the iPaq.
3. By selecting either entry and confirming, Alice can start the desired task.

[8] Computer aided design: www.cad.com

Figure 3.5: The ARCHIE selection menu displayed on the *iPaq*

Scenario: **Calibrating the Devices**

Actor instances: `Bridget:User`

Flow of Events:

1. When she starts the calibration method with her iPaq she also needs to have the 3DOF pointing device in her hand.
2. Bridget can now see the current virtual 3D scene not calibrated on the 2D image plane. In addition to that the calibration scene appears superimposed in her HMD, too. And she is asked to align the peak of the 3D pointing device with the corresponding 2D image calibration point. Once Bridget aligned the points properly, she confirms the measurement by touching her touch pad glove.
3. As the calibration method needs at last six measuring points to calculate the desired projection parameters, Bridget will be asked to repeat the last step for several times.
4. After confirming the last calibration measurement the newly calculated calibration parameters will be transmitted to the viewing component.
5. Now her HMD is newly calibrated and can augment her reality. So the tracked real objects can be overlayed by corresponding virtual objects in front of her.

As the working environment is calibrated and ready for use, two architects want to perform a collaborative task: They want to develop the outer shape of a new building.

Figure 3.6: HMD calibration with a pointing device

Scenario: **Modeling and Form Finding**

Actor instances: `Charlotte, Alice:User`

Flow of Events:

1. Alice and Charlotte start the ARCHIE modeling application and their HMD viewing services.
2. As the system is initialized, both see the environment of the later building site.
3. Alice takes a tangible object, moves it besides another already existing building and creates a new virtual wall object by pressing the create button on her input device.
4. Charlotte takes the tangible object, moves it to the virtual walls position and picks up the virtual wall by pressing the select button on her input device.
5. Charlotte chooses the new position of the wall and releases it from the tangible object.
6. Both continue their work and build an approximate outer shape of a new building.

Scenario: **Mobility - Location Awareness**

Actor instances: `Dick:User`

Flow of Events:

1. Dick starts the location-awareness subsystem based on the ARToolkit [31], running on his laptop attached to his backback. In addition to the former setup an iBot camera is mounted on his shoulder to deliver video images.
2. The system is configured with the current room information. The information consists of the current room and the outgoing transitions (doors) to other rooms. A pie-menu giving information about the current ser-

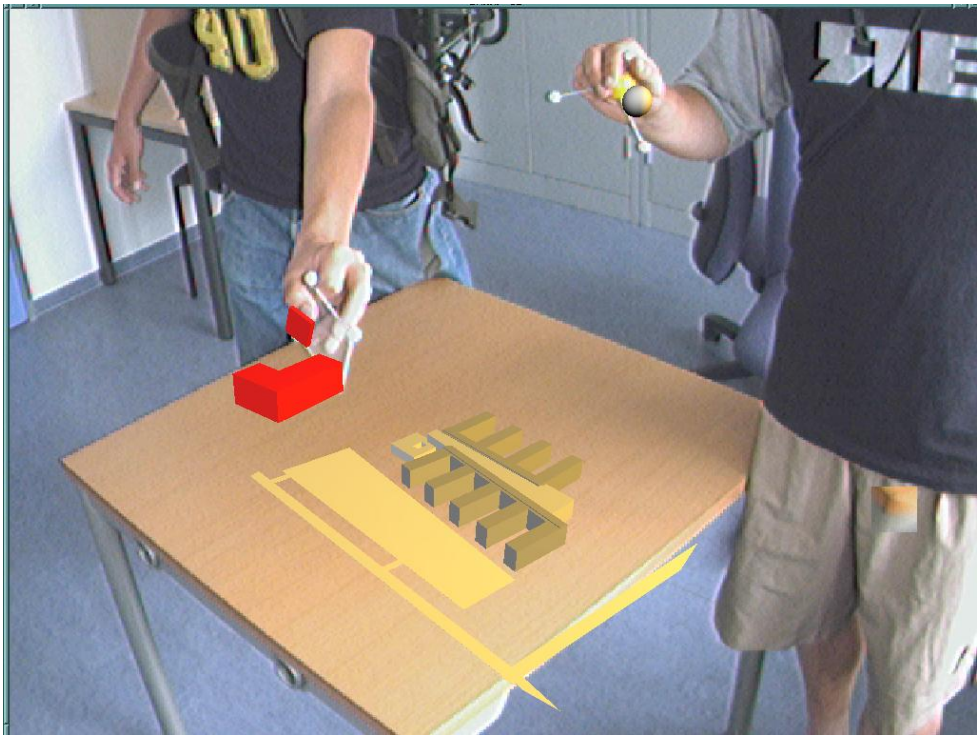


Figure 3.7: Modeling and Form Finding

vices of the room appears on the laptop.

3. Dick exits the room while detecting an ARToolkit marker attached to the door with his iBot camera. The system changes its configuration according to the new state (new room). The old information is dropped and Dick has a new set of services available now which can be used in the current environment dynamically. The pie-menu is updated.
4. Dick is able to exit and enter rooms and is enabled to use the corresponding services and room environment.

After different other Augmented Reality supported tasks are done, a group of later building end users visit the architects office, going to become introduced to the plans of the architects office.

Scenario: **Presentation**

Actor instances: `Alice:User`

Flow of Events:

1. Alice starts the system, but instead of the previous used HMD, now a video beamer view is started, providing scenes as seen from a tangible camera object.
2. Alice takes this camera and moves it around the virtual model of the planned building.
3. The public can get a spatial understanding of the proposed building

Figure 3.8: Hardware setup for the location awareness

which is displayed on the beamer screen. The model shown in figure 3.9 is rendered in anaglyphic red-cyan 3D. For a realistic 3D view the visitors need to wear the corresponding red-cyan glasses.

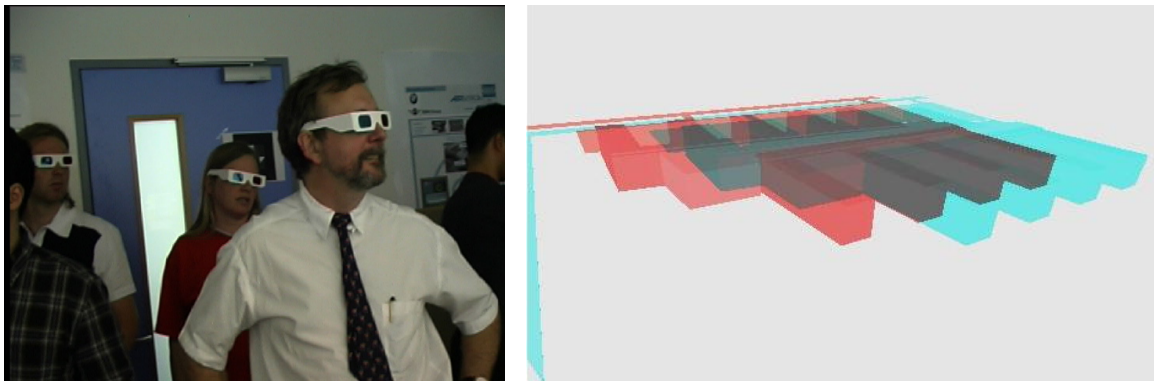


Figure 3.9: Presentation of a planned building to an audience

Scenario: **User Interaction Evaluation**

Actor instances: `Felicia:User, Gabriel:Evaluation Monitor`

Flow of Events:

1. Gabriel starts the ARCHIE application and configures it for an usability evaluation task.
2. He sets up the logging system and initializes it with the study and task name Felicia will be performing for an unambiguous log file.

3. After briefing Felicia appropriately, she is asked to perform a number of tasks which are being monitored.
4. The logging system is automatically taking task completion times while incrementing the task counter too, so Gabriel can fully concentrate on Felicia's reactions to the system.
5. While Felicia is performing her tasks, Gabriel observes a number of real-time, updating charts which visualize her performance by e.g. applying standard statistical functions.
6. During the course of the study, Gabriel is always fully aware of what Felicia is seeing in her augmented display by looking at a special screen which duplicates Felicia's HMD view.
7. Felicia is debriefed after she has completed posttest questionnaires handed out by Gabriel.

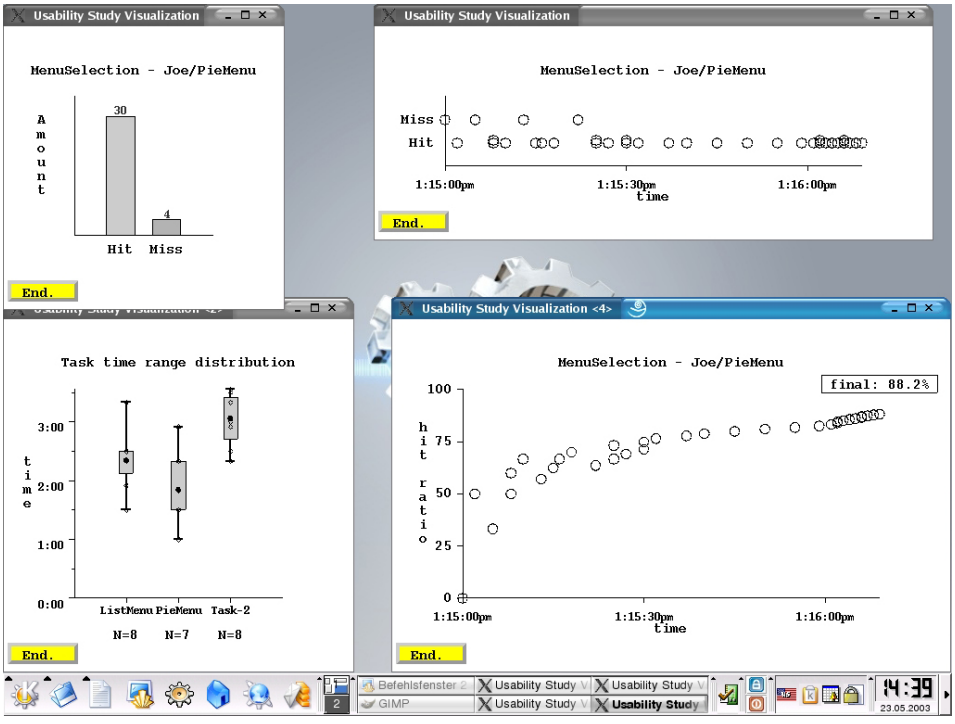


Figure 3.10: Live visualization of user performance in usability study

3.5.4 Requirements

This section describes the requirements our team elicited during the startup phase of the ARCHIE project. The methodology described in [15] was used to refine them step by step.

3.5.4.1 Functional Requirements

Functional Requirements describe interactions between a system and its environment [15]. They are independent from the implementation, but can get realized almost directly in code. This section declares the *Functional Requirements* for the ARCHIE system.

Modeling process

Helping Wizard Architectural applications have many wide ranging functions. Desktop versions provide menus, toolbars and context menus. Entering the Augmented Reality domain will deprecate some functions by intuitive interaction, but still an option for selection of available functions is necessary.

Moving and Placing Objects For a good spatial understanding of virtual 3D-building models, these should be movable in an intuitive way. Rotating and moving functions should resemble the real world physics.

Interrupted Work The application must preserve the modeling states when the users switches back and forth between different applications.

Collaborative work

Shared Viewpoints Shared viewpoints are useful for other passive participants to watch the work of the designer. The system has to support the reproduction of a single view on multiple terminals. For a larger audience a videobeamer view would be even more useful.

Personal Views During the development process one architect might chose one certain submodel for editing which is then locked to his usage until he publishes the edited submodel again for his colleagues to see. So the changes can be be propagated to all participating viewpoints for consistency.

3.5.4.2 Nonfunctional Requirements

In contrast to the *functional requirements*, the *nonfunctional requirements* describe the user-visible aspects of a system [15]. These may not directly relate to the system's functional behavior. *Nonfunctional requirements* can be seen as overall requirements a system must fulfill. They influence multiple locations allover the later realization. In decomposition they can be seen as *functional requirements*, but this view would be hard to understand for the client during requirements elicitation.
This section reveals the *nonfunctional requirements* of the ARCHIE project that lead to the design goals of general DWARF framework components.

Augmentation

Correct in Place Alignment of Virtual Objects In order to have a Augmented Reality viewing device in architectural context, the viewing display must be calibrated so that virtual graphics are rendered in the position and orientation corresponding to the real world.

Three Dimensional Augmentation Spatial impressions of the outer shape of constructions such as buildings require three dimensional views on the virtual model.

Real-time It is a basic principle of Augmented Reality applications that the user can not distinguish between real and virtual objects. To uphold this illusion the view must update rendered objects at a high speed and accuracy so no significant differences can be seen compared to real objects in behavior. Therefore the tracking subsystem should provide adequate precision, too.

Convenient Wearable Devices Because the development of complex buildings is a long duration process, devices must be comfortable to wear and use. System output and input devices such as HMDs and gloves should be attached to the user in such a way as to minimize the loss in freedom of mobility.

Ubiquity

Mobility There could be more than one Augmented Reality enabled office in an architectural bureau, so the users Augmented Reality device should support mobility by wearability and provide the execution of the application wherever possible without requiring additional setup steps. This encourages better collaboration between participants.

Application Selection Dependent on Location Often there are different projects in a company, available only at certain locations. So there should be a dynamic selection of applications and tasks depending on the users current context.

Robustness In addition to omnipresence of computers the system must handle input data gracefully from possibly very large amount of users simultaneously. The system has to differentiate input devices by users to avoid ambiguous data streams.

The development for framework components also yield requirements.

Providing Service Functionality The services provided to the framework should fit in one of the architectural layers specified in the DWARF explaining section.

Dynamic Application Configuration Framework components may rely on context information. These services must be configurable via interfaces as described in [40].

Quality of Service Changes done by a user in a collaborative session must propagate to views of the colleagues. All views within the system participating on the same application need consistency. This aspect also applies to data not directly handled in a view, like internal service configuration.

3.5.5 System Design

An overview over the architecture of ARCHIE is shown in figure (3.11).

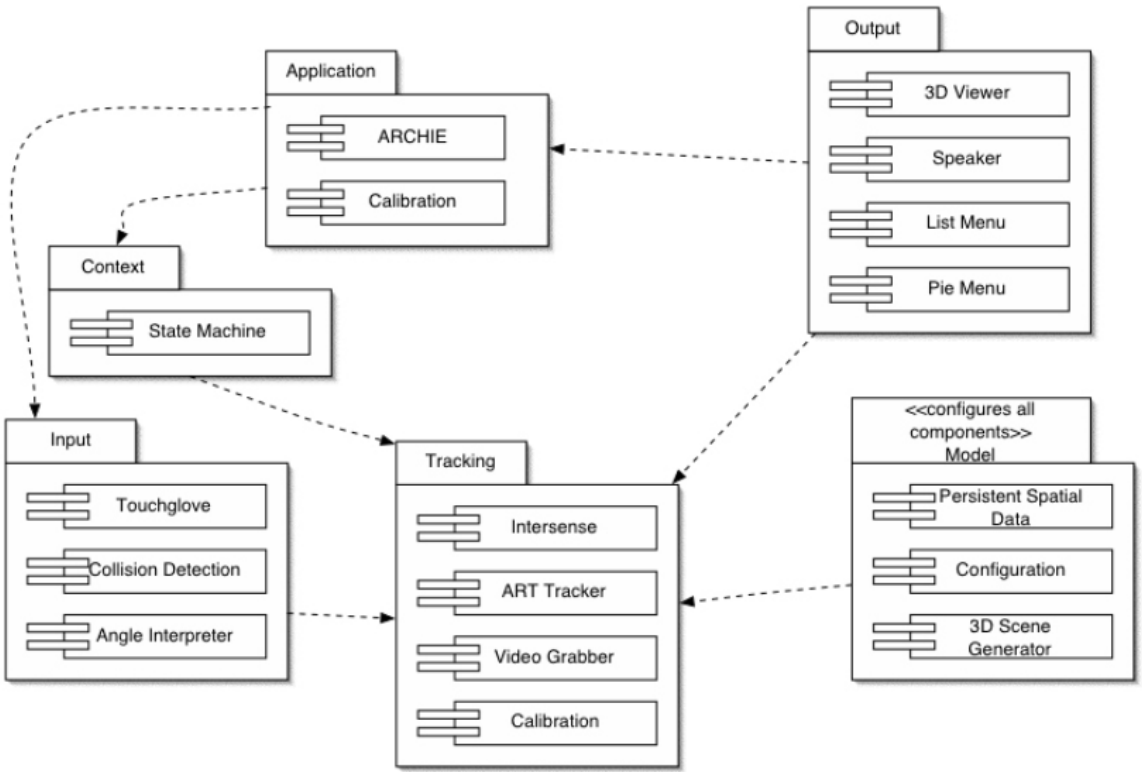


Figure 3.11: ARCHIE architecture

3.5.6 Focused Tasks

A usable framework has emerged from DWARF, but it is yet far from being complete, if one can speak of completeness on such a wide ranged research topic at all. So the implementation of new components must focus on the individual research topics of our team members. This may result in some application specific components, but hopefully they

will get generalized in later DWARF projects.

Since students make up the workforce, who require a precise subject assignment, approximate tasks where given to the majority of our group members from the beginning. So the framework was extended with the realization of narrow topics.
These two restrictions result in the following implementation selection:

Input Device A SEP that enables a touchpad mounted on a glove in DWARF.

Output Device Another SEP provides a 3D rendering service which adapts to different output hardware.

Middleware Improvements Though the service-manager offers a variety of communication and connection interfaces, there is still a need for authentication of identifiable components. This is a SEP, too.

Location Awareness The last SEP within the team improves on optical feature tracking making location awareness possible.

Adjustment of Viewing Devices A diploma thesis that supplies calibration and configuration to different visual output devices on demand.

Managing the Users and Application Context Another thesis adding components to the framework for management of data of different types as 3D scenes and component configuration [66].
The proposed components are intended to hold information about real and virtual objects as well as to hold information about user context specific DWARF service configuration data.

Usability Evaluation of Human Computer Interfaces in Ubiquitous Computing The third thesis in our group provides a usability evaluation framework which can be leveraged to evaluate human computer interfaces within the Augmented Reality system to come to e.g. new Augmented Reality design guidelines. This includes, among others, components to take user performance measurements in DWARF, and visualizing this data appropriately to support the usability engineer monitoring a participant for usability study purposes.

To complete this list we also want to mention the work of our architectural client, who provided architectural requirements. This work has a direct focus on Augmented Reality. The thesis topic contains visualization of various kinds of radiation as warmth flow and refraction. A bargain, DWARF might hopefully provide in the future.

There are other components required for ARCHIE which do not fit in one of the above listed categories. These will get generalized hopefully in new projects.

Although the second part of this chapter focused on the ARCHIE projects, one will see the development of reusable components for the DWARF framework in all cases of our team members in the following chapters.

Part II

Usability Engineering Process

4 Requirements Analysis

The usability evaluation part of the usability engineering lifecycle has to be covered.

This part is about the process aspects of the usability engineering evaluation framework proposed in this thesis.

Since the focus was on the software aspects, core requirements will be listed here directly, while omitting the listing of scenarios and use cases. Please refer to the third part on page 66 for a more detailed requirements elicitation.

Many of the following requirements are a direct result of the intended application setting of this framework: The university it was developed at.

4.1 Functional Requirements

Functional Requirements describe the interactions between the system and its environment independent of its implementation [15].

Since the process does not have functions in the same way a piece of software does, the list for functional requirements is rather short.

Cover Usability Evaluation The aspect of usability evaluation in the whole usability engineering life-cycle is the clear focus of this thesis and as such the framework has to cover this aspect.

Require Only One Person Due to low man-power only one person should be required to do all the evaluation related work. This especially means that this person should not have to rely on other help while actually conducting a study with participants.

Easy Setup It should be straightforward to setup a study once all required materials have been acquired to not miss the chance of evaluating the performance of new participants when they become available on short notice.

Modifiability It should be possible to adjust the framework on demand when new requirements arise in the future.

Utility The framework should be actually usable to gain new insights on usability to e.g. extend the yet naturally minimal design guidelines.

4.2 Nonfunctional Requirements

Nonfunctional Requirements describe user-visible aspects of the system that are not directly related with the functional behavior of the system [15].

These requirements are again mostly resource related.

Good documentation This thesis might be the basis for future work.

Low Monetary consumption Conduction of the process should be cheap to not exceed the limits of the university setting.

Low Time Resource consumption Testing a few participants should be sufficient to get useful results. This is also why the focus should be on quick exploratory studies. On the same vein the process should not ask for too much time from participants so they can still be motivated to conduct the study without any compensations.

5 Related Work

What does the whole usability engineering lifecycle look like and where does this thesis fit in?

Usability engineering has been practiced by many companies for a while now and so much time and effort was already spent in developing and fine-tuning the usability engineering lifecycle.

This thesis is mainly concerned with a small part of this lifecycle, the usability evaluation. However, to fully understand how this fits into the big picture, it is helpful to take a look at the whole lifecycle.

Of course there is no such thing as the one perfect usability engineering lifecycle, so the next section will merely give an overview about one of the more popular ones.

5.1 Usability Engineering Lifecycle

See figure 5.1 for a good overview of one example usability engineering lifecycle [42].

The usability engineering lifecycle consists of a set of usability engineering tasks applied in a particular order at specified points in an overall product development cycle.

The cycle starts with structured usability requirements analysis tasks. Using this data, precise usability goals are set in the next series of tasks.

Driven by these goals and other requirements data, a structured, top-down approach to user interface design is taken.

Finally, objective usability evaluation tasks assist the iterative design process to meet the prior set goals. This part is exactly were the focus of this thesis lies at.

The above can be split into three major phases which are covered one by one in the following.

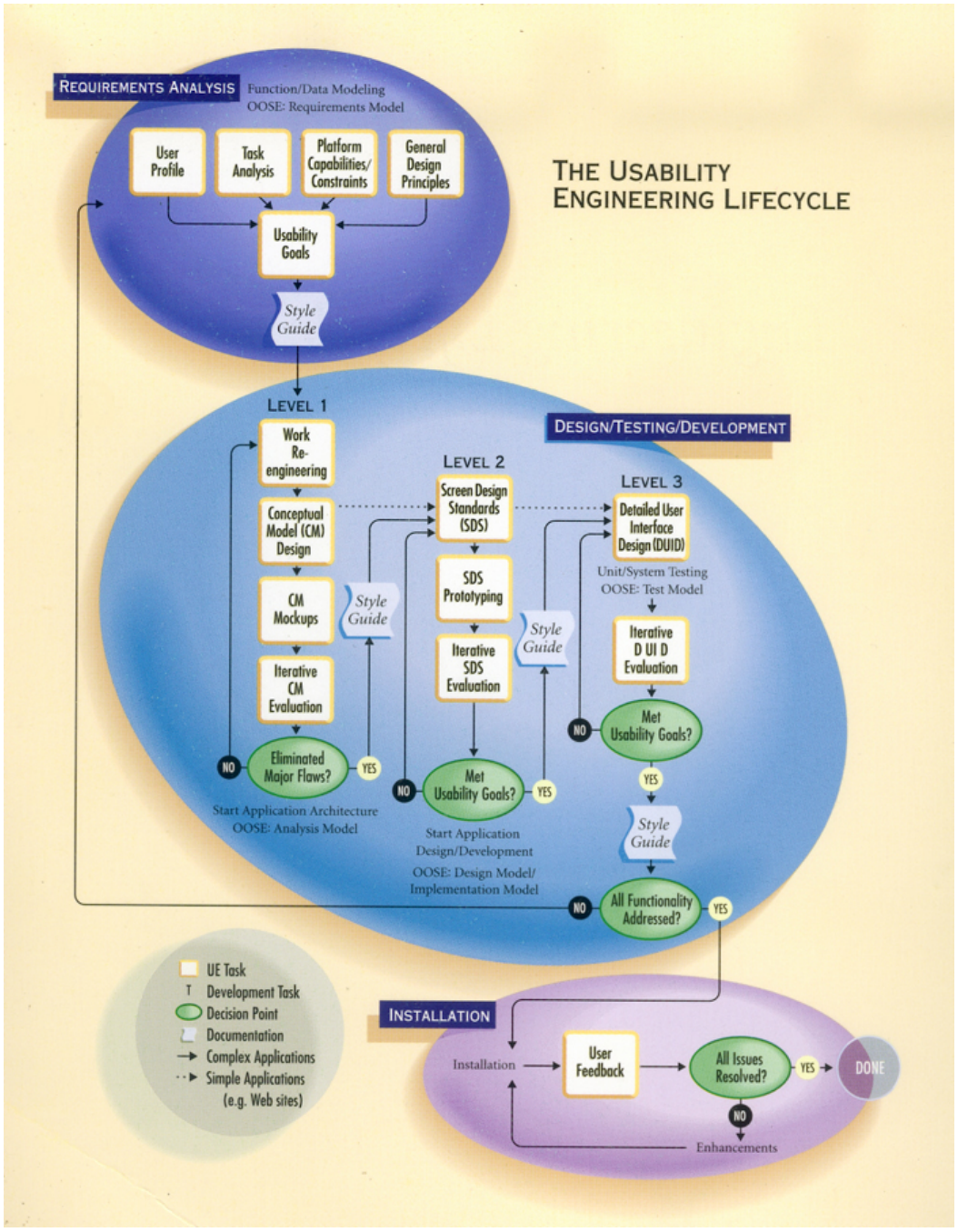


Figure 5.1: Usability engineering lifecycle (Courtesy of Mayhew et al.)

Requirements Analysis

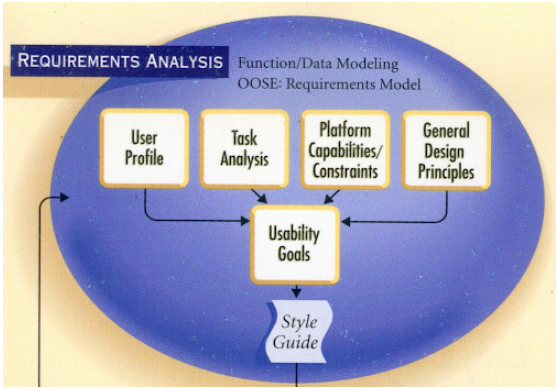


Figure 5.2: Requirements Analysis (Courtesy of Mayhew et al.)

To gather precise, both qualitative and quantitative, usability goals, which are later used as benchmark data to be used in usability evaluations as acceptance criteria, two types of input are taken into account.

User Profile Each product usually has specific target user groups who have specific characteristics. User profiles matching these personas are generated as a reference point for design decisions. This usually takes, among other things, computer literacy and expected frequency of use into account. This could go even so far as to define specific personas like "Mary" who is 64 years old, likes sports cars but can't program her VCR. Later in the design process, it helps to think about features as in "Would Mary really like or use this?".

Conceptual Task Analysis Usually, products are designed to improve an already existing work-flow, since it is rather hard to come up with never before thought of user goals and which nobody has yet tried to achieve.

So here the current work-flow patterns are analyzed to get an understanding on the underlying user goals and to obtain an user-centered model of work as it is currently performed. To achieve this, studies should be conducted in the users' actual work environment. Even filming could be leveraged for task analysis [34].

Often the new product will aim at e.g. reducing the number of steps in the work-flow while still achieving the same user goals.

Task scenarios are generated which can be leveraged to extract, by means of software engineering [15], e.g. functional and non-functional requirements the product must fulfill.

Having now gathered goals defining minimal acceptable user performance and certain satisfaction criteria, these goals now have to be documented in an evolving work product

called “Style Guide” together with platform capabilities and constraints, which result from the chosen technology platform.

The style guide enriches communication regarding UI design across the project teams.

When usability evaluation studies are conducted, guidelines for general, good user interface designs can be collected [60]. Generic design considerations give a rough idea on how to start designing systems. Without applying these first, usability evaluations might never stumble on an usable solution. Relevant literature, if it exists on these topics, is gathered or taken from prior studies to be applied to the further design process.

These general design principles could be also applied by enforcing them with automated authoring tools, which have the guidelines build in.

Design/Testing/Development

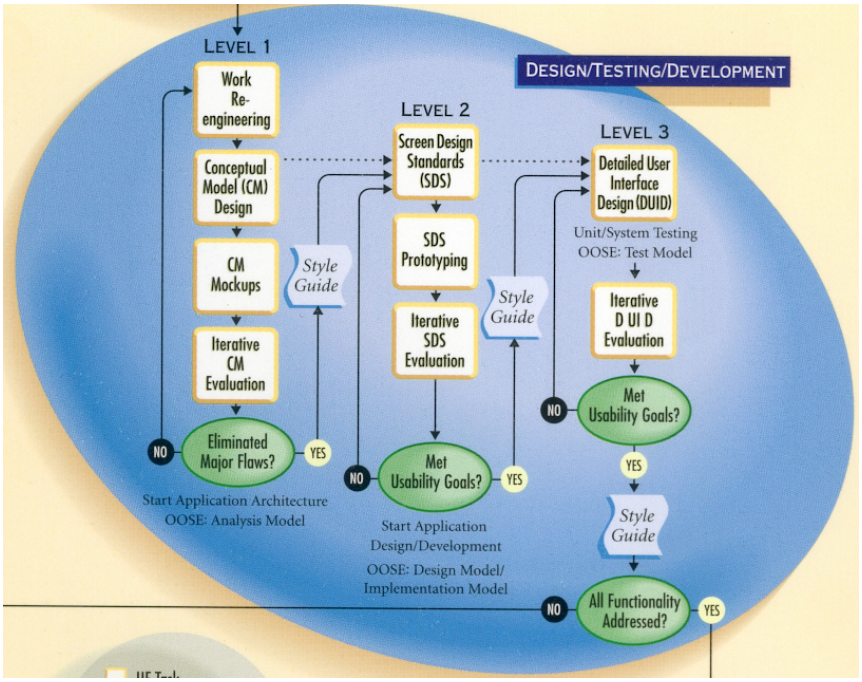


Figure 5.3: Design/Testing/Development (Courtesy of Mayhew et al.)

The bulk design phase is here split into three levels which get more detailed with each level.

Level 1 The first level is concerned with high-level issues only. “Work Re-Engineering” is about streamlining the work and exploiting automation capabilities at an abstract level without involving user interface design, while minimizing retraining and maximizing productivity.

The re-engineering could take place with card sorting techniques to re-shuffle work steps or with general task scenario walkthroughs identified earlier.

With all the current gathered information, navigational pathways and major displays, while looking for a consistent presentation, are identified in the "Conceptual Model Design" phase, generating a coherent and rule-based, high-level UI design framework. Major displays only specify how functionality and information is divided across individual displays between which the navigational pathways make movement possible. Major displays include for example an area to display a construction site, an area for a palette of tools or a help screen. Actions like typically "undo" or "redo" with no feedback in form of a new display area are not considered at this point. This accommodates the users' natural expectation of a unified model across the UI.

Paper-and-pencil or prototype mock-ups are generated from the last phase in "Conceptual Model Mockups" which are the very first products in this lifecycle, with a small subset of total product functionality, which can be tested.

These mock-ups are then evaluated, refined and validated with formal usability evaluation in the "Iterative Conceptual Model Evaluation" phase. Already in this phase representative participants should be gathered to eliminate all major flaws in the conceptual model, before proceeding to the more detailed next level.

The stable results of the conceptual model are captured in an updated style guide. See figure 5.4 for an example conceptual model.

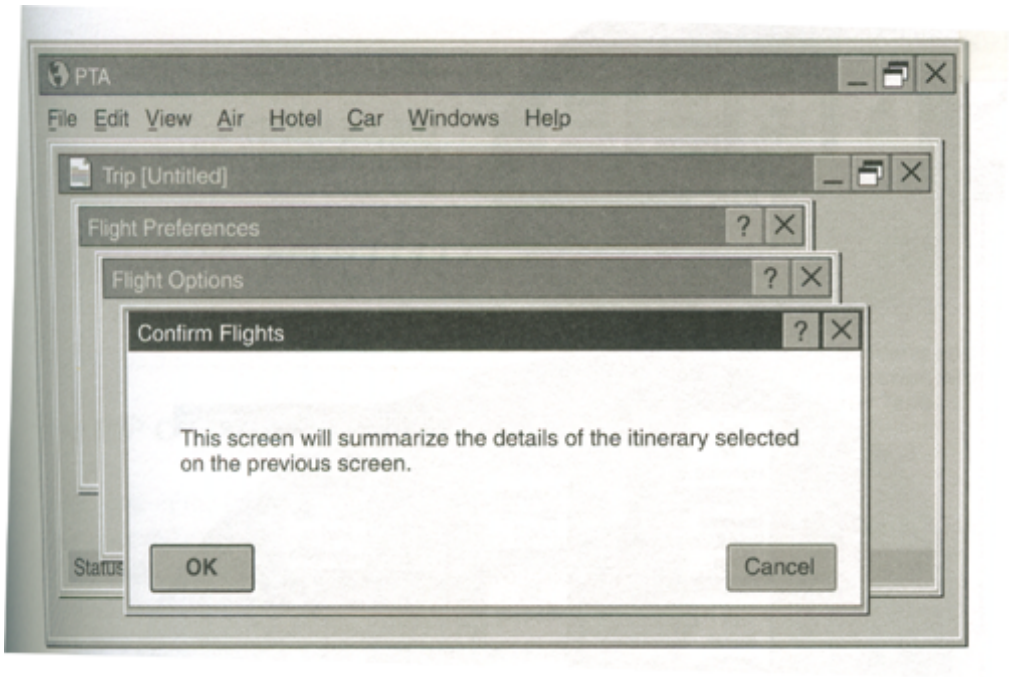


Figure 5.4: Example Conceptual Model (Courtesy of Mayhew et al.)

Level 2 Level two is about setting standards in screen design to ensure coherence and con-

sistency in UI design across the displays and interactions in the product UI.

To define the "Screen Design Standards" all the previously gathered data is combined with product-specific standards or conventions. This should also consider corporate standards which represent the corporate identity so users will recognize the product branding. Screen design standards typically include aspects like color usage, type of navigational controls and other style related questions.

Having defined standards for screen design, first prototypes can be implemented for further evaluation again in an iterative fashion until all major bugs are eliminated resulting in stable screen design standards which are to be noted in the style guide.

These prototypes still have only a subset of total product functionality. This does change in the next level however.

See figure 5.5 for an example screen design mock-up.

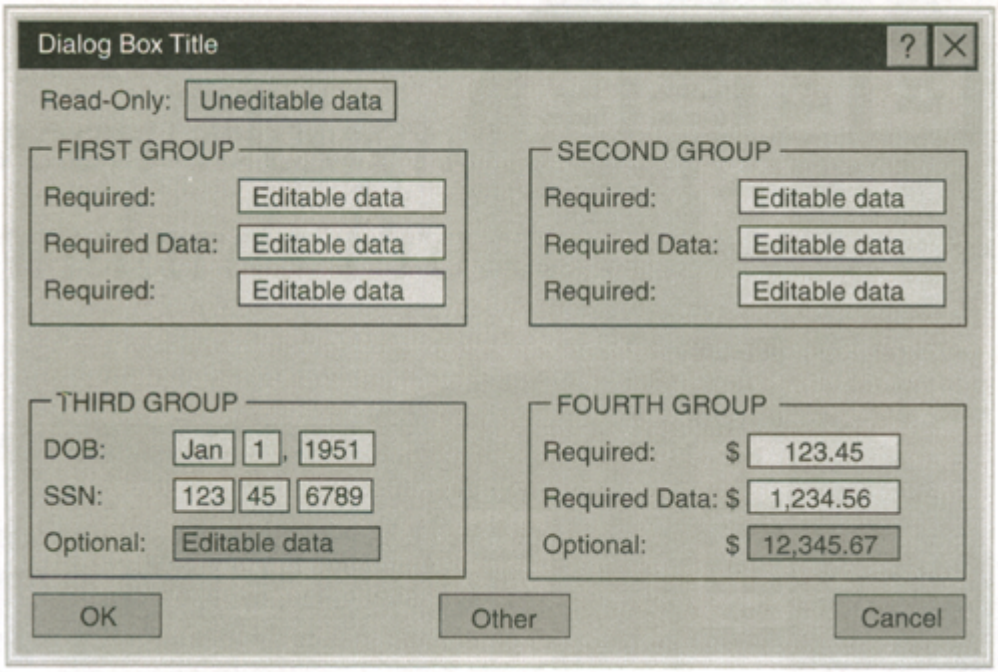


Figure 5.5: Example Screen Design (Courtesy of Mayhew et al.)

Level 3 Driven by the style guide, all remaining UI abstractions are resolved in the final level, "Detailed User Interface Design" resulting in products which are more than ever suited for iterative, this time more detailed user interface usability evaluations. These studies will get to evaluate the complete functionality of the product which has not been implemented up to now.

When it was ascertained that the product fulfills all usability goals while meeting the required functionality, it is ready for release.

See figure 5.6 for an example detailed design.

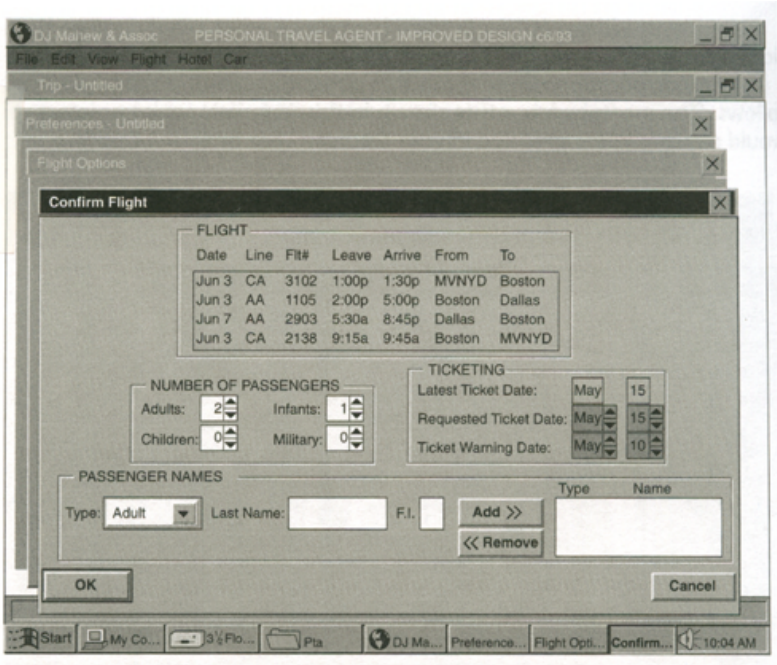


Figure 5.6: Example Detailed Design (Courtesy of Mayhew et al.)

Installation

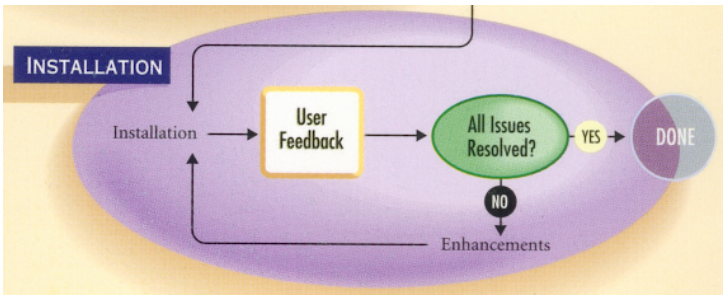


Figure 5.7: Installation (Courtesy of Mayhew et al.)

In this phase the product is actually released to the customer for installation. It still makes sense, however, to observe the product over time collecting valuable user feedback e.g. in the form of questionnaires.

This feedback provides ideas for future upgrades or maintenance aspects of the product, and for the design and development of likewise products which share a similar target audience. Of course this also generates general usability lessons which might be applicable in future efforts like in the leveraging of newly generated general design principles.

Conclusion

There are certain aspects to consider about the above lifecycle.

Adaptation It is important to note that the lifecycle has to be customized on a case-by-case basis meeting the existing environment and cost-benefit considerations. Still all the above tasks apply, but when and how they are integrated with the development process can't be answered in general terms.

The available resources and knowledge play an important role in this adaptation. For smaller projects it is certainly much more appropriate to e.g. collapse the three design levels into one single iterative one, reaching total functionality coverage much earlier than the full rigorous lifecycle does.

Phased Scope Extension In my opinion, it does make a lot of sense to expand the usability evaluation scope over multiple phases. For Ubiquitous Computing and likewise technologies e.g. the rough interaction metaphor ideas should be validated and stabilized as far as possible before evaluating the actual implementations.

However I believe it can be very dangerous to always only look at certain part aspects disregarding possible side-effects between them. The single components might turn out to have great usability on their own, but their combination might cause unexpected usability problems. So the manner of integration between the sub-components is a big determinant of the final usability of the whole system.

This is why the best data for usability evaluations is probably extracted in the final studies where everything is taken into account including the actual work environment.

Chicken-Egg Problem Ideally a lifecycle would be structured in a way so that usability evaluations are obsolete. The lifecycle would consider simply everything and by going through it thoroughly it makes sure that the final product has the perfect degree of usability.

This describes the common "chicken-egg-problem". How can we design a product from the very start which is usable without the need to re-do the design over and over again in multiple iterations due to studies? This is a very interesting question and certainly requires more research than is possible in this thesis.

However, I don't think a solution to this problem will be found ever. The interfaces we talk about are human computer interfaces after all. Maybe we know everything about the computer, but we are certainly way off from understanding the full complexity and diversity of the human brain. Without doubt, by continuous usability evaluations this understanding increases and with time, by applying the previously learned, less and less usability evaluations and thus development iteration cycles will be required to achieve the much desired goal of good usability.

But this has never been so hard as it is now with Augmented Reality and similar technologies since these technologies have capabilities of occupying human senses and using brain regions which have been left untouched in previous human computer interactions.

5.2 Usability Evaluations

Having established that our focus lies on usability evaluations, let's take a look at two different ones. The first is meant to be used in the above shortly introduced complete usability engineering lifecycle, the second one is separated from this.

Integrated in the Lifecycle

Usability evaluations take place in different phases of the lifecycle with corresponding implications.

Conceptual Models When there are only conceptual mock-ups, evaluations are wide and informal. No code has been written yet, so recommended changes are easily realized by creating new mock-ups. Participants are asked to think aloud since there is no need to be concerned about negative performance impacts yet.

These evaluations aim at identifying and removing basic flaws at the level of general user interface standards reaching qualitative usability goals.

Screen Design Standards In this phase prototypes are implemented but still without a real commitment to the code. The code is usually very messy and will be thrown away prior to entering the next phase.

So recommended changes drawn from usability evaluations are still very easy to implement and cost-effective.

The evaluations get more formal and structured, including more detailed tasks covering a subset of the full functionality. Quantitative goals can be measured against with e.g. timing data already.

Detailed Design Evaluations in this phase are done against real final application code with commitment. Tasks are very formal, structured and detailed here focusing on quantitative goals. Users will not be asked to think aloud anymore, to not negatively impact their performance due to this distraction.

If the evaluations were done properly in the earlier phases, only cosmetic problems are expected to be found, which are relatively easy to fix.

However since the scope is widened here to evaluate the complete functionality there is always a small chance that unexpected big flaws come up which might require backtracking to one of the earlier phases. This is however very costly in this stadium and needs to be carefully decided considering cost-benefit analyses.

The actual studies are all similarly structured despite the phase they are conducted in. First the study is to be planned and supportive materials are prepared. After running the actual evaluation in the second phase, the collected data has to be analyzed and interpreted.

Planning and Preparing Not just usability engineers and user interface designers should be involved in the planning phase but also e.g. project managers and developers, in general everyone affected by the outcome of the study. If they were not involved in the planning at all they might want to discard the results.

Decide on Evaluation Type Depending on the previously established usability goals a decision has to be made if to evaluate ease of use or ease of learning. To evaluate ease of learning, minimal instructions are handed out to see how a beginner handles the product. For easy of use, prior training is given to simulate expert usage.

Select User Groups and Tasks Next it has to be decided what kind of user groups should be tested. User profiles have been defined already in the lifecycle in the beginning so it should be straightforward to select groups to evaluate. The plan should include how many users are to be evaluated, and what characteristics they should bring at which skill levels.

What kind of tasks are to be evaluated?

a. Tasks which are very likely to be used by all users.
b. Older updated features, or new yet unevaluated ones.
c. Rarely executed tasks, but which must have good usability, like emergency tasks.
d. Tasks which are of special interest to marketing.
e. Tasks which the team has general concerns about.

There usually is not enough time to evaluate all tasks at once, so a decision has to be made.

Design Tasks Tasks can be compared on two dimensions. On the one dimension we have "results based" and "process based" tasks on on the other "dependent" and "independent" tasks.

Results based tasks, define a goal to achieve for the user like "Construct a new simple building". Here the user is on his own and will have to find the correct steps to achieve this goal. There might be multiple ways to achieve this goal so you will never know in advance which path is taken by the user resulting in possibly hard to compare data. However it is a good approach to find out which path is the easiest one to find. "Process based" tasks enforce a certain step sequence by splitting the bigger overall task in several smaller ones which set a specific path. For example steps for the above task could be "Start the modeling application", "Place the bottom floor on the construction area", "Add four walls on the sides", and "Add a housetop".

Dependent tasks build on each other, so failing an earlier one, the user gets stuck and might need help. Usually these task scenarios are more realistic than totally independent tasks with no connection whatsoever. For the user it is easier to follow a realistic scenario, than to execute a number of tasks which seem to have an arbitrary order which might confuse the user. However it is hard to design an usability evaluation which is supposed to evaluate a number of completely independent features of the system with one logical connected scenario.

Usually one participant is evaluated after the other but sometimes it is beneficial to compose teams of participants to evaluate e.g. collaboration features of a new product.

Design Evaluation and Evaluation Materials A single session for one user should take between one and three hours to collect enough data while not being too tiring.

There is a number of materials which are to be prepared for the studies.

- *Observer Briefing*, instructions to anyone who wants to observe the evaluation. Usually it asks the observer to remain quiet and not to interrupt during the evaluation.
- *Welcome and Training*, short statement of appreciation to read for the participant or which is to be read to her.
- *Pretest Questionnaire*, used to gather demographic data on the participants and to verify the matching user profiles.
- *Permission/Non Disclosure forms*, these forms signed by the participant allows e.g. filming and prevents information disclosure.
- *Test Task Listing*, often handed to the user with task descriptions. For complex tasks these are handed one at a time.
- *Data Collection Tables*, papers with prepared tables to allow quick written notes during the study.
- *Posttest Questionnaire*, handed out to the user after the finished task to gain subjective opinions.

To conduct the study, an evaluation environment has to be set up, usually with video cameras. Finally participants are to be recruited through various means. For formal evaluations about three to ten representative people should be won.

To verify the above material and that no major bugs remain, a pilot study should be conducted before moving on to the actual study.

Conducting the Evaluation The study is made up by a number of standard steps which will be further elaborated in Chapter 6.

To quickly summarize, the above prepared material is handed out in the logical sequence while observing the participants, taking notes and trying not to influence them in any way.

Analysis and Interpretation To begin analysis a few topics have to be regarded.

Participant Inclusion After all participants have been evaluated the data analysis can start. Usually this is done across all participants, but sometimes it makes sense to do this for specific subgroups if they are different enough, e.g. when participants were recruited for multiple different profiles.

Collation All data is then summarized and collated for example by number of errors per task, error types, number of users experiencing specific problems or amount of time to complete tasks. Percentages could be e.g. computed to show how many participants performed successful within a certain time benchmark.

There are a number of possible visualizations or cumulations for the collected data.

Inferential Statistics Often Inappropriate In most cases, simple data summaries are sufficient. Inferential statistical analysis is usually not appropriate in an engineering environment ([42], [56]). The number of evaluated participants is typically too small and mandatory assumptions for statistical analysis, like normal curves or random sampling, are not satisfied. Additionally both evaluation monitors and the audience for the test results are often not familiar in interpreting inferential statistics properly.

Pareto Charts Let's assume the data analysis identified a usability problem with printing. Multiple reasons for this problem were found and participants got stuck at different ones, e.g. printer off-line, printer cable not connected, document sent to wrong printer etc.

A good summarizing method here are Pareto charts which group these problems [42]. The left-side vertical axis of the Pareto chart is labeled Frequency (the number of counts for each category), the right-side vertical axis of the chart is the cumulative percentage, and the horizontal axis of the is labeled with the categories. See figure 5.8 for an example. This form of visualization is well suited to answer questions like which 20% cause 80% of this usability problem (80/20 Rule)?

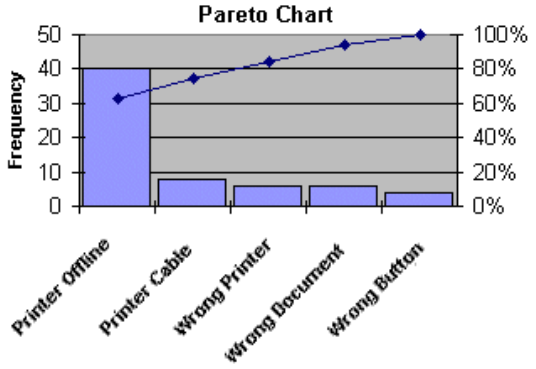


Figure 5.8: Example Pareto chart

Impact Analysis Tables Another means to weight relative severity of identified problems is by impact analysis tables ([26], [42]). Usually here the time which is spend in error is measured or estimated and then subtracted from the actual task time to estimate the time the task would take if the problem was fixed. Putting this data in a table format will then show which of the errors had the highest quantitative time impact.

Mean Scores To calculate the mean task completion time a simple calculation of $Mean = \frac{Sum\ Of\ All\ Participants'\ Task\ Completion\ Times}{Number\ of\ Participants}$ is executed [56]. However if e.g. only one out of six participants had a task completion time which was four times larger than all others, taking a median score would yield a more typical result.

In any case though, the border values, that is the whole range, should be closely watched since their evaluation might crop interesting insights.

Median Score If all values are sorted, the median score is the middle one which is a typical result even when there have been very extreme values towards both ends [56].

Standard Deviation The standard deviation is a measure of variability, showing to what degree values differed from each other [56]. It takes into account all values.

It is calculated with $Standard\ Deviation = \frac{\sqrt{\sum x^2 - \frac{(\sum x)^2}{n}}}{n-1}$

where n is the total number of values and $\sum x$ their sum.

The result should be put relative to the mean score. If it is relative small, all values were clustered pretty close together with the opposing argumentation accordingly.

Anova Sometimes multi-way ANOVA equations are calculated to disprove or prove prior well defined hypothesis by means of statistical significance with a large number (usually 30 ore more) participants in extremely well controlled evaluation setups [25].

The above listing cannot be exhaustive and only provides typical ones, since the best type of visualization is very dependent on the inspected task and as such they should be chosen on a case by case basis by applying common sense. Of course "simple" measurements such as times could always be visualized with mean and standard deviation for decent results.

After data has been analyzed conclusions and recommendations for design changes should be put together. Often these are given priorities depending on their usability impact and their cost for implementation.

Conclusion

My research has shown that usability evaluations are always executed by the same patterns. This makes sense since there is not that much room for radical improvement here. Instead the process has to be adapted to be feasible in the current environment. This is done in the following chapter 6.

I do agree with the notion of [56] and [42] that inferential statistics such as Anova calculations are inappropriate in the setting of usability evaluations. Especially so with the pragmatic approach with few participants I decided to take for my sample study as described in part IV.

6 Usability Evaluation

Which adaptations are required to standard usability evaluation processes to meet our requirements?

This chapter briefly outlines the process for usability evaluations. I found that Rubin [56] provided a good generic framework which I adapted according to our circumstances.

In each section I will first give a summary of Rubin's [56] process, followed by the adaptations I have made. If there were no adaptations required the detailed descriptions have been omitted. Please refer in these cases to [56]. Some of these points have also already been covered in the previous chapter 5.2 and are not repeated here either.

6.1 Four Types of Evaluation

Summary According to Rubin [56] there are four types of usability evaluations with different implications.

Exploratory This type of evaluation is very informal, often on an abstract level, and happens very early in the development cycle usually working on the conceptual model to eliminate major design flaws.

Assessment This typical study is conducted early to midway through the product development cycle. This is basically a screen design evaluation which tries to interpret how effectively the conceptual model has been implemented. While intuition is a big focus in exploratory studies, more low-level questions are raised here.

Validation Validation evaluations are on the detailed design with final code usually set against a number of predetermined usability benchmarks measured often in user performance. Only when it is validated that the product meets these benchmarks it is ready for release. This is also often the first real integration evaluation because every component comes into play at full functionality.

Comparison The comparison evaluation can be combined with any phase. Combined with an exploratory study, different conceptual mockups could be compared, combined with assessment the effectiveness of different implementations of a single feature could be inspected. Finally combined with validation, the product could be compared to similar, competing ones.

Performance and preference data is collected in the side-by-side comparison. The level of formality can be set freely from less formal explorative studies to highly structured experiments with control groups, focusing on one single dimension, to achieve statistically valid results with a pre-defined hypothesis with expected results.

Adaptation This differentiation is very similar to Mayhew [42] as in having a different focus on quantitative and qualitative measurements and a increasing level of formality depending on how late the evaluation occurs in the lifecycle.

For the university setting, researching Ubiquitous Computing technologies, I believe a less formal exploratory/assessment comparison study makes most sense. Ideally authoring tools for Augmented Reality technologies should be available to quickly put together different interfaces as prototypes for a cost effective quick evaluation with few participants.

In my sample study outlined in part IV, I focused more on qualitative statements than on quantitative ones. I always asked all participants to "think aloud" to get as much qualitative feedback as possible, while still measuring task completion times to draw even more qualitative statements from.

The human computer interfaces of Augmented Reality technologies are still widely unresearched with many doubts on basic levels. We are still on the level of technology demos far away from any really usable systems. Proposing a hypothesis and conducting a tightly controlled study with a high number of participants seems unreasonable considering these basic uncertainties.

Concluding, I see a high need for explorative studies evaluating the intuitiveness of conceptual mock-ups of Augmented Reality type systems. Sometime in the future enough results will be available answering the core questions, allowing reasonable formal validation studies concentrating on fine-tuning the system. The previous generation of human computer interfaces, GUIs, is today based on the WIMP paradigm which is founded on years of research. I can envision finding a similar paradigm for Wearable Computing in the future which will be the base to fine tune on. Exactly this is what explorative studies are perfectly suited for.

6.2 Evaluation Environment

Summary Usability evaluation usually takes place in a specialized room, the usability lab. Ideally you do not want to use a lab for usability evaluation but rather monitor the user at his actual workplace thereby enabling the monitor to take the whole context into account.

Practically this is hard to realize though and not feasible until after fine tuning was done in a lab.

Rubin [56] defined four different laboratory setups going from cheap and simple to expensive and elaborated.

Simple Single-Room Setup The setup is shown in figure 6.1.

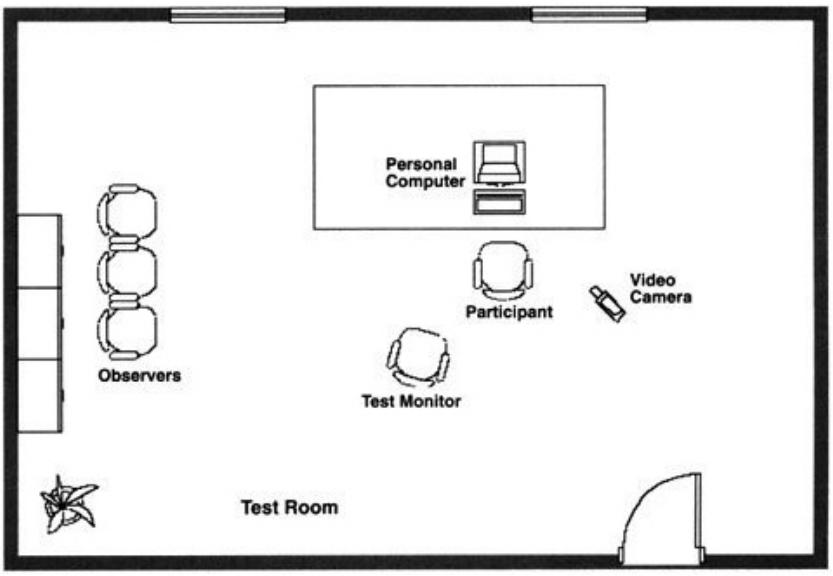


Figure 6.1: Usability Evaluation Simple Single-Room Setup (Courtesy of Rubin et al.)

Both space and resource requirements are minimal in this approach. The monitor is sitting at a distance of about four to six feet at about 45 degrees to the participant to still have a good view while not irritating her. The monitor has to avoid giving even subtle clues to the participant which is easier when he isn't sitting up very close.

Easy interaction with the participant for explorative evaluations is possible while providing an optimal view to the action.

However there is very limited space for observers and the risk of influencing the participant with unconscious subtle reactions on behalf of the monitor is very high. Great care is needed by the evaluation monitor to not bias the participant by e.g. shifting the posture or taking notes at a bad moment which might steer the participant into a different direction. The evaluation monitor is in a conflicting position. On the one hand he has to be as close as possible to gather every possible subtlety which might be important such as minimal gestures from the participant but on the other hand he shouldn't steer the participant in any direction which the participant wouldn't have taken on her own at that exact moment to not invalidate the evaluation study.

The setup is easy enough to even allow "mobile laboratories" where the necessary equipment is just brought to a suitable room.

Modified Single-Room Setup As seen in figure 6.2 the distance between monitor and participant is increased here with help of video monitors while still being in the same room.

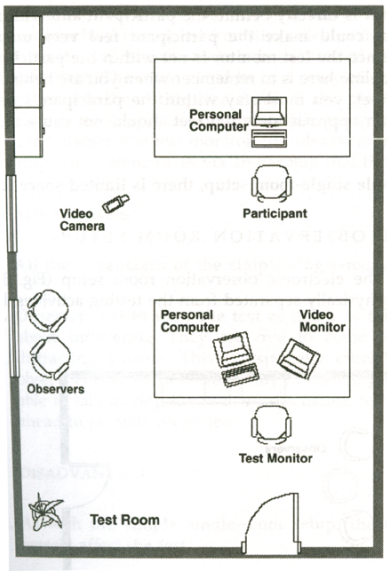


Figure 6.2: Usability Evaluation Modified Single-Room Setup (Courtesy of Rubin et al.)

The main advantage here is that the participant is less likely to notice subtle unintended monitor behavior. However this of course also limits as to what can be seen by the monitor without an excellent camera setup.

Electronic Observation Room Setup This setup is very similar to the simple single-room setup but with an added observation room specifically for observers. If desired, this allows totally non-distracting observation since they are in effect in a different room.

Classic Testing Laboratory Setup The most expensive setup has a special evaluation room where the participant is sitting an a control observation room. The participant is isolated in the evaluation room while everyone else is in the control room behind a one-way mirror. Microphones and speaker systems are used for communication.

There are no biasing effects at all possible by the monitor unless he makes a mistake on the intercom. Loud communication is possible between observers in the control room.

However this setup is very impersonal and might make the participant overly self-conscious, sometimes requiring the evaluation monitor in the evaluation room after all. There are only little interaction possibilities plus the cameras have to be set up very well to not miss important clues.

Adaptation For our purposes a simple single-room setup which could theoretically be set up at any place without too much effort (Figure 6.3) requiring the least amount of resources and space is the most reasonable.

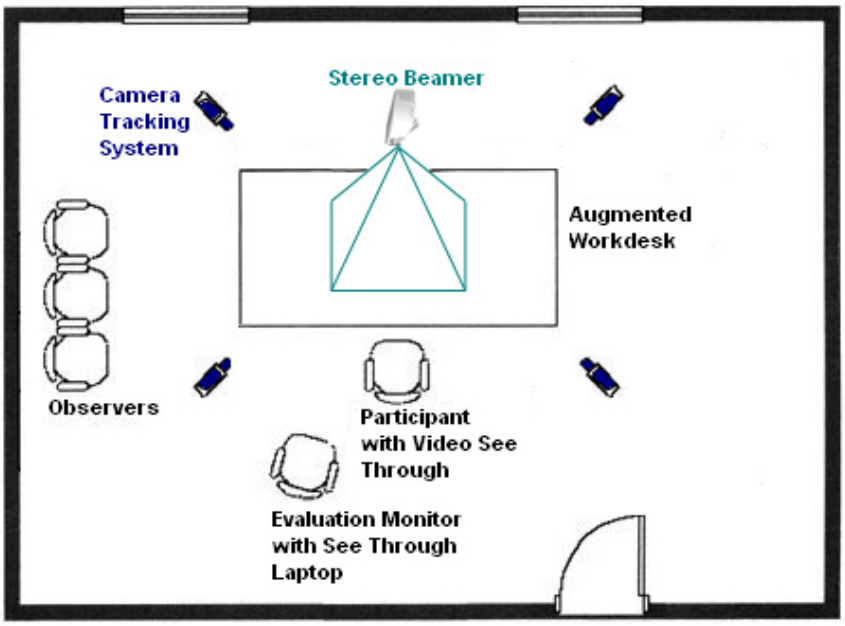


Figure 6.3: Evaluation Room

This setup is well suited for explorative/assessment evaluations too.

Augmented Reality requires some form of projection and tracking devices, for example realized by a stereo beamer and a camera ART tracking system which take quite a lot of effort to be set up however. Because of this circumstance it's practical to only conduct the usability evaluation studies within a specialized lab within the university which is configured to support augmented reality at all times.

A stereo beamer is projecting the augmented area on top of the desk the participant is working on. The participant is wearing a head mounted display (HMD) which has video-see through capabilities in stereo mode. The participant sees a stereo live video image with stereoscopic virtual objects superimposed on it [33].

The evaluation monitor is again located about four to six feet from the participant at about a 45 degree angle for the above mentioned reasons.

The monitor has a laptop with him which displays the exact same live video-stream the participant is seeing at this very moment in his HMD plus a mask to enter comments and other short important data. The video-stream is logged on disk to be reviewed later by the monitor. Additionally the laptop is capable of operating in see-through mode should the monitor wish to switch perspective on the augmented scene.

Observers may optionally take place at the scene but they must be far enough away from the main scene to not irritate or influence neither the participant nor the monitor. To view the augmented scene they should be provided with standard see through laptops.

In the sample study we didn't actually have or use a stereo beamer but a totally different setup which required yet another modified layout for the evaluation room as shown in

figure 6.4 as a mock-up, or in figure 11.11 on page 127 in real-life.

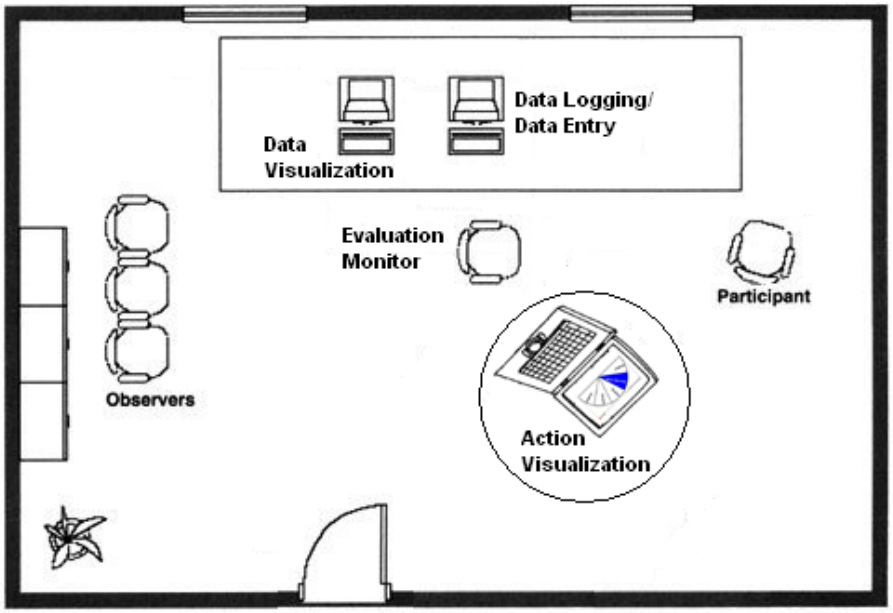


Figure 6.4: Sample Study Evaluation Room

The participant is again wearing an HMD which is attached to a laptop, which renders the scene simultaneously both on the laptop TFT, as well as in the participant's HMD for live action visualization. The monitor is keeping the above mentioned distance while having a good view on the user. The computers to the left of the monitor are used for live performance data visualization and manual log taking. The user's wearable devices are attached to one of the computers on the left.

6.3 Different Evaluation Roles

Summary Rubin [56] defined a number of different evaluation roles. At a professional setup there are usually multiple people involved in conducting a usability study.

Evaluation Monitor This is the most important role who has the responsibility of overseeing the whole study and assembling all the results. At the very least he will do all the interactions with the participant and compile the results.

Ideally this person is a human factors specialist with a degree in psychology or similar. He doesn't have to understand the technical implementation details of the evaluated product at all. Often times it is even beneficial if he is out-sourced from the company, so he is not emotionally attached with the product. This way he can give really objective results.

Data Logger This person is merely responsible for logging all data by e.g. checking of marks at a data collection sheet.

Timers Similar to the above, this role has a stop watch and takes task completion times of the participant.

Video Recording Operator All the video recording is handled by this role.

Product/Technical Expert If technical issues come up during the evaluation which is not unlikely early in the development phase this role handles technical aspects such as resetting the system in case of error.

Adaptation All of the above roles can be handled by one person and that's the approach I propose in the university setting to save resources.

As described in the software part of this theses (Part III on page 66) automatic tools handle time taking which free the monitor to focus on the action at hand. I learned that the fully automated time taking certainly eased my task as an usability monitor since I only implemented it after the pilot study. Another tool offers a simple data entry mask to quickly take notes which are to be reviewed later. Since the monitor is usually sitting close to a computer to watch the live action anyway, it does not take much more effort to quickly type in some notes on this same computer.

Video recording does not make much sense in our environment since the review of this video data takes a lot of effort and time. But this is not really required anyway due to the simple single-room setup. The monitor should catch every subtlety on the live run if he is just watchful enough. If it is desired, a camera could still be set up at a fix location which is expected to show most of the user's actions but this is not very easy with a participant who might move around with a wearable device.

Usually at the university setting, implementation of the user interfaces is done by the same person who will conduct the studies which has been the case in this thesis, meaning the monitor is already also the product and technical expert.

6.4 Stages of Conducting an Evaluation

Usability evaluations usually follow a standard pattern already quickly introduced in chapter 5.2.

6.4.1 Evaluation Plan

Summary Having an evaluation plan for the study is important as a guideline to follow [56]. The plan is serving as a template to fill with data as the study progresses.

The plan should include the evaluation purpose, the problem statement, the user profile, the test design, the task list, list of all required resources, the role of the evaluation monitor, evaluation measurements and finally the results.

Most of these aspects have already been covered in chapter 5.2 or did not need adaption so we will look at the test design options [56].

Independent Group Design When the sets of tasks to be evaluated are very large and or to completely avoid learning in between task sets, different groups are used who only carry out one of the task sets.

Within-Subjects Design Independent Group Design of course requires quite a lot of participants. In "Within-Subjects Design" the same participants perform all task sets. To avoid learning in between them, counter balancing is used where the order of task set presentation is balanced.

However this can get difficult when the task sets are logically depended on each other like already mentioned in 5.2.

Counter-balancing can also be applied when evaluating multiple product versions while having multiple participant groups [56].

Of course there is no maximum limit for participant numbers. The higher the number the more likely all usability problems are found plus statistical analysis starts to get more feasible.

For a true experimental design, at minimum of 10 to 12 participants per condition must be utilized [56]. Research has however shown that evaluating four to five participants already reveals 80 % of the usability problems [56]. It does take much more than that to find the remaining 20 % however.

Adaptation For our environment a within-subjects design makes perfect sense. This requires the least amount of participants while still avoiding learning effects to a certain degree with counter balancing.

I did evaluate nine participants in my study which I consider well sufficient. I doubt the study would have yielded more results than it has (Chapter 11.3.2) if I had evaluated more, although I did gain a little more insights with every added participant.

The main usability problems certainly have been clear well before testing the ninth participant much more around participant four.

Please refer to Chapter 11.3 which describes the actual study methodology I used.

6.4.2 Evaluation Participant Selection

Summary Ideally usability evaluations should only be performed with participants representing the end users. If we were to e.g. design a product which is meant to simplify VHS recording for elderly technological challenged people it would be disastrous to perform all tests with IT savvy computer scientists. Studies like that wouldn't tell us much about how the real target group would perform so it wouldn't be very helpful.

Internal testers are people of the company who developed the product, so they are often both familiar and emotionally attached to it. Still they can be useful for pilot studies, or for

best case evaluations. Having only internal testers will yield very non representative results of course. Each study should have at least one least competent user [56].

In any case it is important to properly screen the participants by asking them about past experiences or letting them perform quick tests to judge their level of competence to be able to later assess the study results.

There are many ways of acquiring participants among many others these are employment agencies, market research firms, existing customers or college campuses.

Adaptation I thought "hallway testing" was most appropriate for the university setting. After I had set up all the laboratory and prepared all the materials, that is questionnaires, I simply evaluated anyone who walked by and could afford the time.

Since these people went by anyway there is also no need to compensate them usually.

Of course this resulted in mostly internal testers but I was still very positively surprised how much usability problems can be uncovered with internal testers. This is of course very task dependent however.

When counter balancing with two sets of tasks is used, I think not only one but two least knowledgeable users should be acquired, one for each task set sequence. I actually acquired three such least competent users and their input has been very valuable indeed.

So although lots of usability problems can be uncovered already with internal testers in my experience a few least competent users should be evaluated too. The sample should also at least include one participant from each gender. If I had not done this my sample study would not have shown that the attachment of an input device near the breast area might indeed pose a problem or annoyance to female users.

6.4.3 Evaluation Preparation

Summary The questionnaires and other material required were already listed in chapter 5.2. To re-cap, the most important ones were "Welcome and training", pretest questionnaire, permission/non disclosure forms, test task listing, data collection tables, and posttest questionnaires.

Adaptation Due to the written software tools, no data collection tables were required at all. The video permission or non disclosure forms are not required in the university setting at all.

I learned that it is a good idea to offer both categorized answers as well as free text ones in the posttest questionnaires since some users will want to fill the free text fields while others wont. It was also interesting to offer a final question about the confusion on the questionnaires to get feedback on that. I consider it important to ask the user about his initial impression on the task difficulty level in the pretest questionnaire and ask him about his revised opinion again after he has experienced it in the posttest questionnaire.

I did not actually need a test task listing since the sample study was easy enough that general usage guidelines (See A.3.2) were sufficient.

6.4.4 Evaluation Conduction

Please refer to Chapter 11.3 which describes the study setup I used which is also applicable in general terms.

Part III

Software Engineering

7 Requirements Analysis

We need data logging and visualization plus action visualization.

This part is about the software aspects of the usability engineering evaluation framework proposed in this thesis. In the following scenarios are provided which describe possible applications of the evaluation framework and its behavior in a concrete situation. Use cases go on to generalize this behavior leading to functional requirements.

7.1 Scenarios

A scenario is a concrete, focussed informal description of a single feature of the system from the viewpoint of a single actor [15].

Here I will describe the scenarios I used when designing the software part of the framework for usability evaluations in Ubiquitous Computing. All of them are actually fully supported, otherwise the sample study described in part IV would not have been possible.

The first scenario shows how the system supports the logging of data. We refer to data as measurements of user performance.

> *Bob, a usability engineer monitoring a usability study, wants to log data to measure certain aspects of user performance. Before starting to log the performance of Alice, he sets up the logging system with initial data, like the name of the study, Alice's name and the first task name. Doing so he assures that he will have unambiguous log data later on which he can map to specific evaluations. Since Alice is performing a rather critical task repeatedly, Bob needs to focus on her reactions all the time. To not be unnecessarily distracted, he wants to setup the logging system in a way so it will automatically take task times and record all events accordingly without further assistance from his side. However when Bob notices that Alice did something unexpected, he wants to take a note, which should be logged together with all the other data. Sometimes Bob is performing studies where the tasks have been put together in a rush so there was no time for setting up automatic task time taking. In these cases Bob has the need to take the time manually with the system.*

The next scenario is complementary to the above one. It shows how the visualization of the data logged above is performed.

Before actually starting the study, Bob thought about good means to visualize the measured data to show possible deficiencies. He wants to put something together quickly, while relying on some previous work he has done on visualizing. While Alice is performing her given tasks, Bob wants to see the graphics he thought about, live while they update themselves. Combined with the facial and otherwise expressions from Alice he always has a clear picture on what is going on during the task. After Bob is done with all participants, he needs to put together a report for Steve the responsible manager in "AR-Construct", the company which asked him to evaluate the product. He remembers that Alice has demonstrated a very serious flaw in the design in her second task by looking through the notes he has taken earlier. Bob prints the corresponding visualization in a high quality version to be included in the report.

Finally for proper monitoring, the system should support some form of live action visualization. "Action" refers to whatever the user sees like e.g. the actual 3D scene augmented in a real construction site while she is performing the task given to her.

Bob notices looking on his live updating visualizations that Alice is performing suddenly very poorly. Alice also seems very annoyed and starts swearing about the new system from "AR-Construct". She is wearing an HMD which augments her reality by rendering the building she is trying to re-construct using her Data-Gloves in front of her. Only by looking at the screen which shows to Bob what Alice is seeing, he can correlate her poor performance to a previous error. Alice turned the dial for zooming in accidently a bit too hard before and now sees nothing but a big white wall in front of her. She must have forgotten what this dial was for.

7.2 Use Cases

A scenario is an instance of a use case, that is, a use case specifies all possible scenarios for a given piece of functionality [15].

Here I present the use cases extracted from the above three big scenarios.

Data logging

The first scenario was about logging data and can be split up into the following use cases.

Use Case: **Log data**
Initiated by: `Evaluation Monitor`
Communicates with: `Participant`
Flow of Events:
1. *(Entry condition)* None
2. The `Evaluation Monitor` is installing the system for logging data.
3. *(Exit condition)* The system is now ready to log data.

Use Case: **Define Task**

Initiated by: `Evaluation Monitor`
Communicates with:
Flow of Events:
1. *(Entry condition)* The logging system is installed.
2. The `Evaluation Monitor` is defining the tasks for the coming usability study in the logging system.
3. *(Exit condition)* The logging system now supports "Logging data automatically" for the prepared study.

Use Case: **Logging data automatically**
Initiated by: `Evaluation Monitor`
Communicates with: `Participant`
Flow of Events:
1. *(Entry condition)* The logging system is installed.
2. The `Evaluation Monitor` is starting the system for logging data while telling it to increment the task counter automatically.
3. The `Participant` is performing a single task multiple times.
4. *(Exit condition)* The system takes the time of all finished tasks automatically, and increments the task counter for each iteration to log measurements accordingly.

Use Case: **Enter Manual Log Data**
Initiated by: `Evaluation Monitor`
Communicates with:
Flow of Events:
1. *(Entry condition)* The system was started and the main data logging component is running.
2. The `Evaluation Monitor` loads up the mask for the data logging system.
3. The system responds by presenting a form to the `Evaluation Monitor.`
4. *(Exit condition)* The logging system now accepts initializing data or any other form of manual log entry taking.

Use Case: **Setting up the logging system**
Initiated by: `Evaluation Monitor`
Communicates with:
Flow of Events:
1. *(Entry condition)* The system was started and the main data logging component is running together with the entry mask.
2. The `Evaluation Monitor` fills the form, by entering the name, study and initial task name for the current study.
3. *(Exit condition)* The system receives the initializing data and logs all following events accordingly until new initializing data is received which is accepted at any time.

Use Case: **Taking notes**
Initiated by: `Evaluation Monitor`
Communicates with:
Flow of Events:
1. *(Entry condition)* The system was started and the main data logging component is running together with the entry mask.
2. The `Evaluation Monitor` decides it is worth to log something.
3. Using the mask, the `Evaluation Monitor` enters a note for later review while specifying the type of note he is taking.
4. *(Exit condition)* The system receives the note and logs it persistently for later review with the correct type.

Use Case: **Taking time manually**
Initiated by: `Evaluation Monitor`
Communicates with: `Participant`
Flow of Events:
1. *(Entry condition)* The system was started and the main data logging component is running together with the entry mask.
2. The `Participant` is starting one of her tasks.
3. The `Evaluation Monitor` uses the mask to start the task timer.
4. The system responds by displaying the current running time and offering a reset option to the `Evaluation Monitor`.
5. The `Participant` finishes her current task.
6. The `Evaluation Monitor` stops the timer using the system while specifying the state in which the task was finished.
7. *(Exit condition)* The system stops the timer and writes the task finish time together with the state persistently.

Data visualization

The second scenario about visualizing data previously logged, can be split up in the following use cases.

Use Case: **Visualize data**
Initiated by: `Evaluation Monitor`
Communicates with:
Flow of Events:
1. *(Entry condition)* None.
2. The `Evaluation Monitor` installs the visualization tool.
3. *(Exit condition)* The visualization tool is set up and ready for operation.

Use Case: **Create new visualization**

Initiated by: `Evaluation Monitor`

Communicates with:

Flow of Events:
1. *(Entry condition)* The visualization tool was set up.
2. The `Evaluation Monitor` creates a new visualization and adds it to the system.
3. *(Exit condition)* The visualization system now supports a new type of interpretation with the desired type of visualization.

Use Case: **Life performance visualization**

Initiated by: `Evaluation Monitor`

Communicates with: `Participant`

Flow of Events:
1. *(Entry condition)* The system was setup and is in the process of measuring data and writing it down to a log file. At least one visualization was previously created and added to the installed visualization tool.
2. The `Evaluation Monitor` instructs the visualization tool to show live, updating plots.
3. *(Exit condition)* The system responds by rending plots, which update themselves automatically depending on the measurements of the performance of the current `Participant` using the supplied created visualization instructions.

Use Case: **Off-line performance visualization**

Initiated by: `Evaluation Monitor`

Communicates with: `Management, Design Team`

Flow of Events:
1. *(Entry condition)* A log, containing data of a previously performed usability evaluation exists. At least one visualization was previously created and added to the installed visualization system.
2. The `Evaluation Monitor` uses the system off-line to create high quality charts of certain performance aspects he is interested in.
3. The system responds by generating high-quality charts like requested.
4. The `Evaluation Monitor` puts together a report to pass to `Management` who will in turn update the `Design Team`
5. *(Exit condition)* High quality charts are printed in the report, generated leveraging the system.

Action visualization

The third scenario is about visualizing the action on screen and can be summed up in the following use case.

Use Case: **Life action visualization**
Initiated by: `Evaluation Monitor`
Communicates with: `Participant`
Flow of Events:
1. *(Entry condition)* The `Participant` is performing a task in an ubiquitous environment.
2. The `Participant` is using some form of device which augments his view, invisible to bystanders who lack special means.
3. The `Participant` is doing something interesting to the `Evaluation Monitor`.
4. *(Exit condition)* The `Evaluation Monitor` is using the system to see what the `Participant` sees in real-time.

All of the above is shown in the following UML use case diagram in an overview (Figure 7.1).

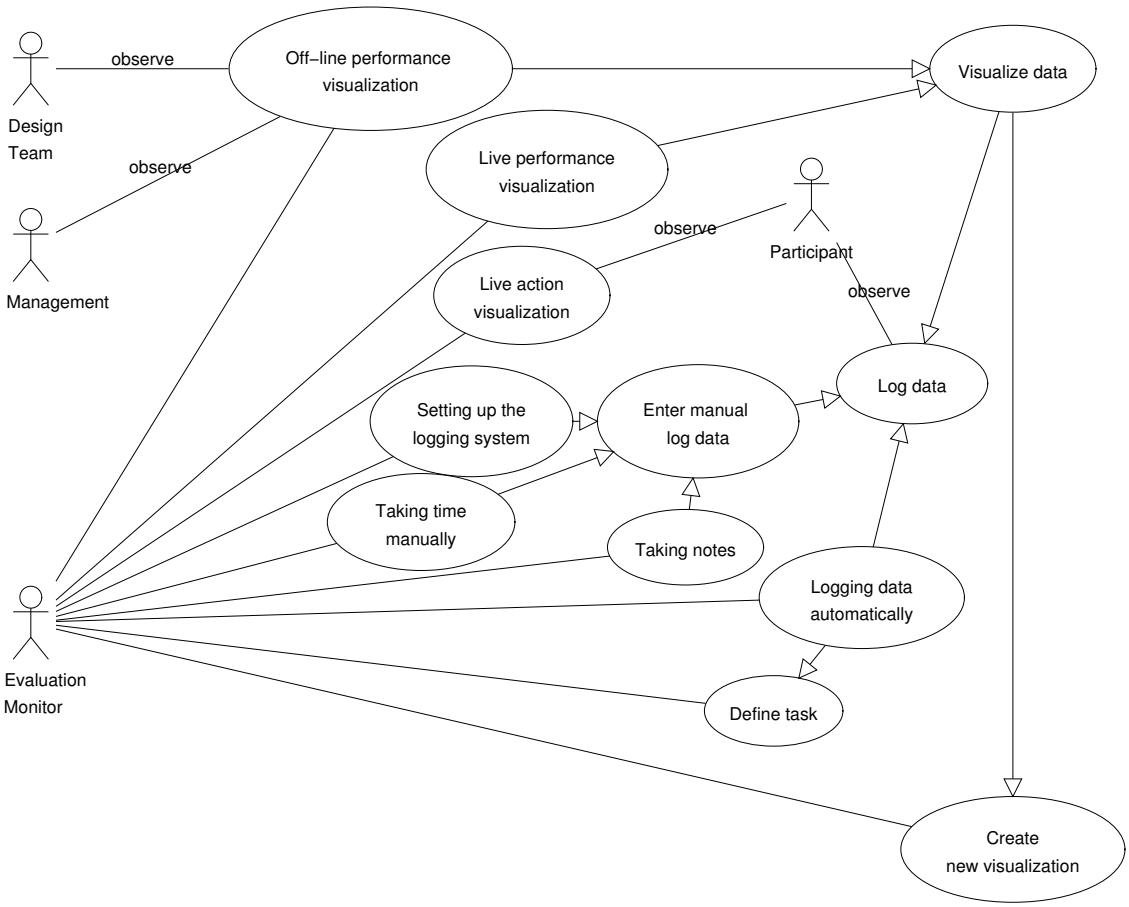


Figure 7.1: UML use case diagram

See figure 7.2 for an activities diagram showing dependencies and the required start order of the use cases.

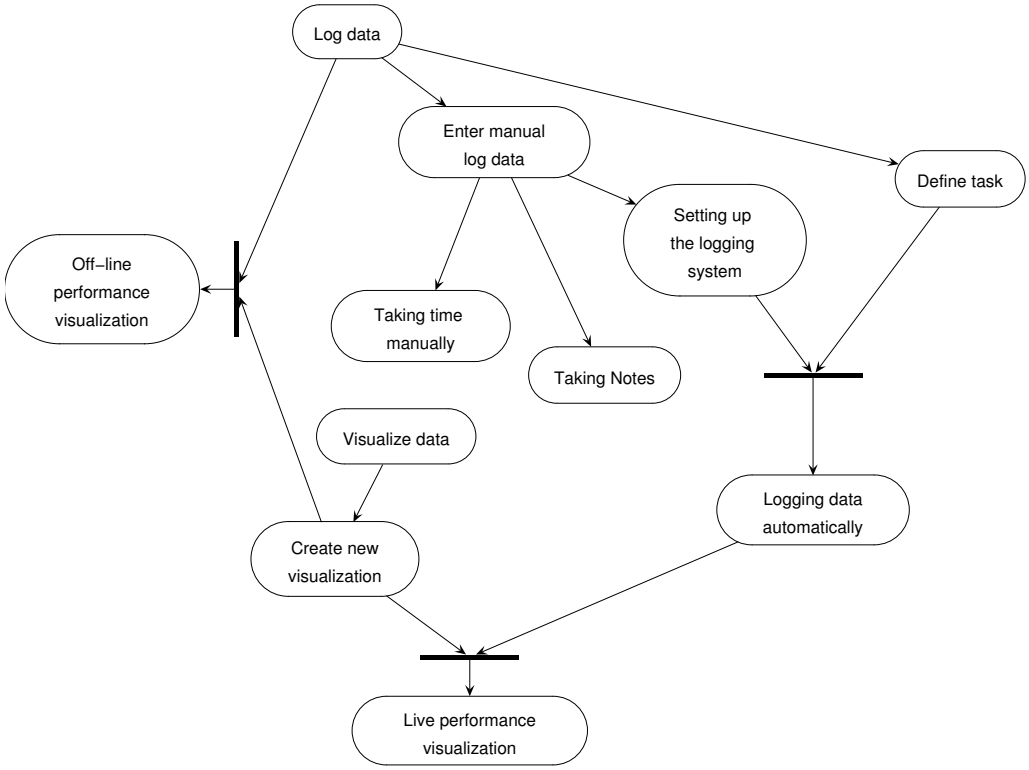


Figure 7.2: UML activities diagram

7.3 Functional Requirements

The functional requirements are consequently also split in the three above groups data logging, data visualization and action visualization.

Data logging

Without data logging in a running system, many aspects of usability evaluation are not possible. We do not want to just rely on black-box observation and questionnaires to perform our studies, so data logging must exist, meeting certain criteria:

Distributed System Support In Ubiquitous Computing there are no monolithic cores to which to attach a data logger. These systems are distributed and thus the data logging system has to cope with this fact by merging into the distributed system, in effect becoming a part of it.

Persistent The logged data has to be stored persistently in a way, that data visualization can take place meeting its requirements.

Unobtrusive/Transparent The logging system must unobtrusively log data, that is appear transparent to all other services which are being logged. No modifications of other services should be required for the logging to work. Additionally the logging must not have any negative impact on the user's experience.

Flexible Little effort should be required to log new measurements or to stop logging ones, which are not any more necessary.

Automated Logging should take place with no user intervention at all if this is desired.

Manual entry support The evaluation monitor should be able to enter special log data to e.g. take notes or to take task time manually during a participant observation into the same logging system.

Data visualization

Solely logging data is not sufficient to draw conclusions from. Only when the data can be processed and prepared in a suitable format of visualization, this can work. This visualization of data has to meet the following:

Rapid Prototyping The data visualization should allow rapid prototyping to be able to quickly generate visualizations, which meets the previously determined requirement. This also means existing data visualization efforts should be re-usable when new visualizations are required.

Statistical Functions The data visualization solution should offer standard statistical functions for log file analysis.

2D Plotting The visualization solution has to offer 2D plotting to draw standard diagrams including bar-charts, pie-charts, line-plots, and range-bars with little effort.

Real-Time Visualization should be possible real-time with "live" data, during a running usability evaluation to assist the monitoring usability engineer.

Unobtrusive/Transparent The data visualization process should have no negative performance or other side-effect on the remaining system.

Off-line High Quality Off-line plotting with high quality output (true type) for inclusion in e.g. paper reports should be supported by the visualization software.

Action visualization

It is very important for the monitoring usability engineer to see what the user sees during his task performance. Only then connections can be made between momentarily user performance and the running program.

Ubiquitous Computing poses new problems to this visualization of performed "action".

Mobile support Users might be walking around in a mobile setup which must be handled appropriately.

Overlay Considering Augmented Reality, means for overlaying a camera picture with the augmented objects are required for proper observation of context.

Persistence It should be possible to store this visualization persistently for later review when required. This property does have a low priority though considering that making use of this feature considerable lengthens the amount of time required by the monitoring usability engineer for off-line review, which is in conflict with the requirements described in part II.

7.4 Nonfunctional Requirements

Data Measurement Changeability Other studies will have very new measurement requirements. Therefore it should be possible to painlessly change the measurements of the logging system.

Data Visualization Changeability It should be easy to change the visualization of data measurements since other studies will have other requirements.

Robustness The logging system and its entry mask must be robust shielding from user input errors.

Performance The software components should allow evaluations without slowing down the remaining system. The mere fact of conducting a usability evaluation should never have any noticeable performance impact on the user experience.

7.5 Pseudo Requirements

Pseudo requirements are requirements imposed by the client that restrict the implementation of the system [15].

The only pseudo requirement was that the logging solution had to be based on DWARF, since this is the framework of choice for Augmented Reality application in the environment this thesis was written in.

8 Related Work

Others have already thought about requirements analysis issues. There already exist third-party tools we could use, with slight modifications, for data visualization.

This chapter covers related work for the software aspects of the proposed usability evaluation framework. It is structured by the three main functional blocks which were identified in the previous chapter, that is "Data Logging", "Data Visualization", and "Action Visualization.

8.1 Data Logging

As Brian W. Kernighan and Rob Pike put it [32]

> *As personal choice, we tend not to use debuggers beyond getting a stack trace or the value of a variable or two. One reason is that it is easy to get lost in details of complicated data structures and control flow; we find stepping through a program less productive than thinking harder and adding output statements and self-checking code at critical places. Clicking over statements takes longer than scanning the output of judiciously-placed displays. It takes less time to decide where to put print statements than to single-step to the critical section of code, even assuming we know where that is. More important, debugging statements stay with the program; debugging sessions are transient.*

There are multiple ways on how to measure data.

Low-Level Operating System Event Logging

In the mobile capture experiments by Kent Lyons and Thad Starner [38] the logging system was put into a very low layer next to the operating system.

Summary The underlying windowing system (X11-Windowing system) was augmented to log all events generated by its application clients and interface devices. This allows evaluations of legacy and or unmodified systems.

Internal Structure A server is responsible for collecting all events and adding timestamps to them finally writing them down to a capture log (Figure 8.1).

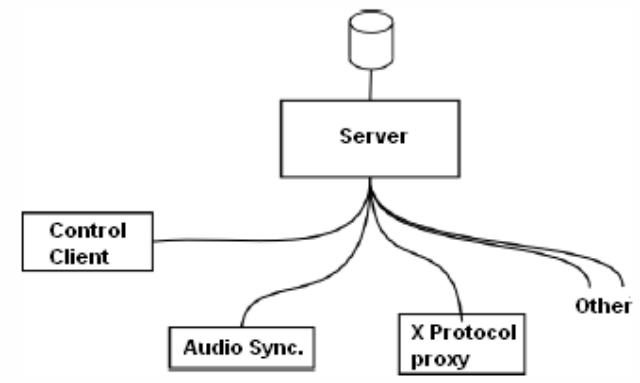


Figure 8.1: Server architecture for logging (Courtesy of Starner et al.)

A set of clients gathers the logging data and propagates it to the server for aggregation.

Control Client This client merely tells the server when to start and stop logging.

Audio Sync. This client controls a custom circuit to achieve video synchronization. Even if all camera recorders and the logging system are started at the exact same time the clocks still drift relative to each other. By using a dedicated audio based synchronization channel, logged data and video streams are synchronized to the same time after all recording was completed.

X Protocol Proxy To gather all possible raw data, the complete X-Protocol was captured. It contains all information exchanged between the X Server and the X Client applications. This is done by this client.

Theoretically this architecture allows custom clients to be integrated which add real semantic information in the future.

But unless this is done, the information stream is very flat with no semantic information at all shifting all the workload for analysis to the visualization tool. For example when a key is pressed the log file would include information on the key-code and other cryptic information like on the states of currently active windows. This data has to be interpreted to extract the actual key and the name of the window which was focused while it was pressed. So the log-file itself is not human readable without post-processing.

Application Code Log4j Type Logging

Summary Log4j is an open source project allowing the developer to control which log statements are output with arbitrary granularity. It is fully configurable at runtime using external configuration files. [5]

Internal Structure Log4j is made up of three main components, loggers, appenders and layouts.

By combining these components, developers can chose to log messages according to message type and level, and to control at runtime how these messages are formatted and where they are reported.

By usage of a logger hierarchy easy inclusion and exclusion of specific data streams can be achieved. For example one main logging hierarchy could log system calls while another one only logs user calls. To even further differentiate the logging stream, levels can be assigned to the different loggers which are "DEBUG", "INFO", "WARN", "ERROR" and "FATAL". Custom levels are usable on demand, too.

An output destination is called an *appender* in log4j. Currently the output of the logging data can be directed to console, files, GUI components, remote socket servers, JMS, NT Event Loggers and remote UNIX Syslog daemons. The appender also supports asynchronous logging.

Finally to customize the output format, a *layout* is associated with an appender, which is based on macros for conversion patterns similar to the C language `printf` function. Special renderer can be defined and registered for often needed logging events.

Of course all code to initiate the log4j logger must be manually placed into the to be evaluated program source code at suitable positions.

Conclusion

The first low-level approach is very generic and can be applied to any AR-type application based on X-Windows which are without doubt quite a few.

The main advantage of this approach is that to compare applications relying on the X-Server, studies can start immediately. Big log data volumes will be generated in both cases for later detailed review. No extra setup is required with no worries that anything could be missed.

Although I agree this has advantages I think the disadvantages are just too great to make this approach really feasible for the following reasons.

X-Server Although many applications finally rely on a X-Server for display, not all do. But then again this last dependency on the operating system can never be removed.

No Semantics The log file data is totally separated from the application and as such totally flat. There is a very big effort required to post-add semantics later.

Semantics Lost Clearly semantics are lost unrecoverable without logging in the application itself at all. Internal happening which might be very interesting for logging can't be captured with this approach.

Huge Log Files Everything is logged. Actually I think this has many positive aspects as this gives the option of later visualizing some measurements you didn't think of earlier. However visualizing tools take a big performance hit when they have to analyze such huge amounts of data when real-time visualization is desired. This was clearly not a requirement of this approach however.

Concluding I think the approach of offering a very generic framework for taking usability data is noble but I highly doubt its utility for actual studies. Using simple visualization scripts in combination with smart logging seems like a more reasonable approach than to do dumb logging in combination with highly complex visualization tools.

The log4j idea promises to help reduce the volume of logged output and to minimize the cost of logging. This is definitely very interesting for usability evaluation related logging since the expected data volumes are very high. By using appropriate layouts, the output could be formatted for further post processing by visualization scripts, thereby speeding up the interpretation process due to the already filtered data, by means of the logger hierarchy. The appenders add the potential of streaming the logging data simultaneously to completely different outputs again increasing the flexibility for data visualization than what is possible with a flat file approach. Integrating these technologies is future work.

8.2 Data Visualization

These tools are about visualizing the logged data or action. There are tons of data analysis and visualizations tools out there already[1]. Usually these tools are not usability specific but meant for general scientific data visualization. Some of the most interesting ones are introduced here.

VizWear

Summary This tool [38] visualizes the interaction information of an event log and relates it to the captured video data.

Internal Structure The application runs on Linux. See figure 8.2 for its interface.

[1] http://sal.kachinatech.com/D/1/index.shtml

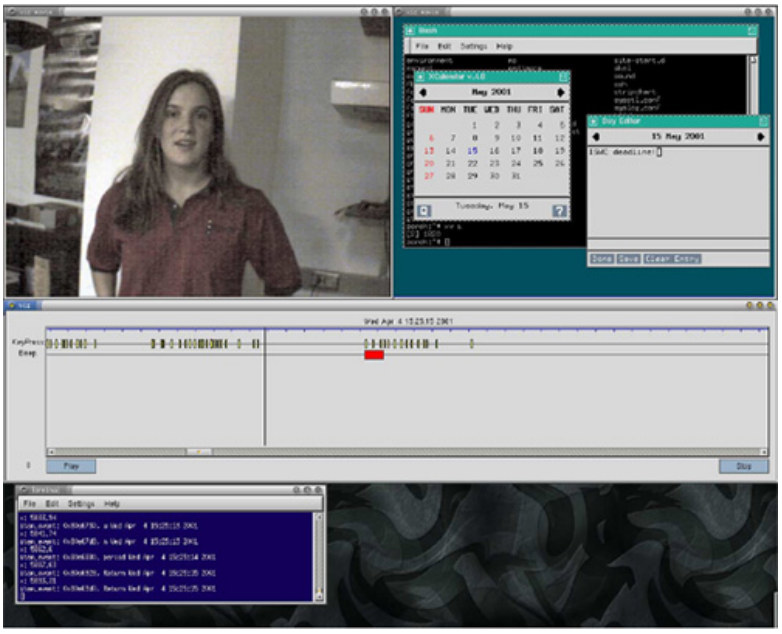


Figure 8.2: VizWear

The tool is meant for data examination and post-trial analysis with the following components.

View of Camera The upper left shows the view of the camera which was placed on the user's head.

View of HMD The upper right shows the video of the view in the HMD.

Timeline In the middle a timeline shows a listing of all logged events synchronized to $\frac{1}{30}$ of a second.

Detailed Log The bottom window shows all log event details.

Scrolling in the timeline will jump to the correct points in the two shown video streams.

The timeline visualizes events in two forms, points and intervals. A point is shown as a vertical line and used for events with no duration like key-presses. Superordinate events, recognized by smaller start and end events are visualized by rectangles showing their full duration. The rectangles could e.g. show the time it takes a user to move the mouse between two elements of a calendar.

Finally hovering over lines or rectangles will show data like the name of the pressed key in the bottom window.

Using the scrollbar or the "Play" and "Stop" buttons further playback controls are added.

Ploticus

Summary Ploticus [4] is free software that generates plots and graphs from data. The primary components are the ploticus script interpreter program "pl" and the libploticus C language API. See figure 8.3 for a short demonstration of what ploticus output can look like when elaborate scripts are used.

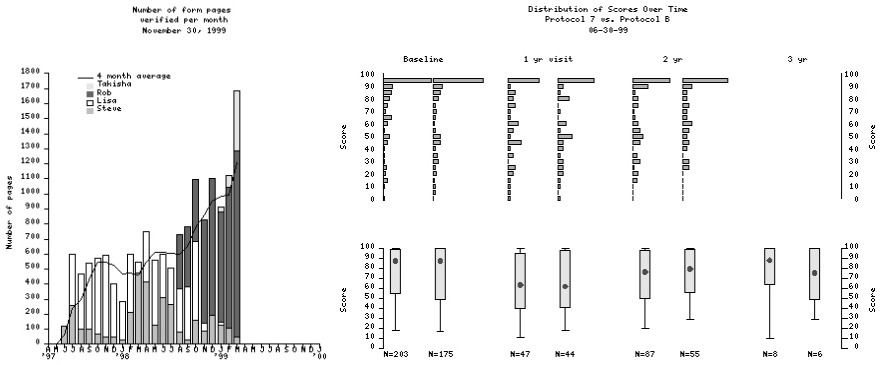


Figure 8.3: Ploticus Example Visualizations

Internal Structure The functionality of ploticus can be best explained by looking at the core aspects.

License Ploticus is licensed under the GNU General Public License (GPL) meaning its source code is available.

Input Basic plots can be created quickly from the command line using pre-build scripts or customized scripts can be written for maximum flexibility. So ploticus has always the need for two different inputs. A script which tells the interpreter how to analyze the second input, the actual data which is expected in the form of tabular ASCII data.

The script file contains settings for the size of the plotting area, defines the title, axes and where to get data, and tells what type of plot to draw.

Output Formats It supports the generation of different output formats including PNG, GIF, SVG or JPEG. PostScript and EPS are supported for paper reports and posters. Graphs may also be viewed interactively on X11 displays.

Plotting Style Support Ploticus supports out of the box all standard 2D plotting styles including line plots, filled line plots, range sweeps, pie graphs, vertical and horizontal bar graphs, time lines, bar proportions, scatter plots in 1D or 2D, heat-maps, range bars, error bars, and vectors. While doing so it supports automatic scaling of axes to accommodate varying input data. Finally legends, annotations, click-maps or mouse-over support can be added.

Statistical Capabilities Numerics, alphanumeric categories, dates and times (in a variety of notations) can be plotted directly. There are capabilities for curve fitting, computing linear regression, and Pearson correlation coefficient r. There is a built-in facility for computing frequency distributions. Means, medians, quartiles, standard deviations, etc. can also be computed out of the box.

Non Interactive The focus has been to develop an engine that can produce graphics non-interactively, so that it can be run in an automated, unattended way.

Ease of Learning Usability During quick experiments with ploticus I could plot graphs with the intended output in a very short time frame. Since I had no prior experience with this tool, I rate its ease of learning usability very high.

gnuplot

Summary Gnuplot is a command-line driven interactive function plotting utility. It was intended as graphical program which would allow visualization of mathematical functions and data.

Internal Structure The functionality is again explored by looking at the core aspects.

License Gnuplot is copyrighted but can be freely distributed as Freeware. Its source code is available.

Input In batch mode it accepts a file containing special gnuplot commands. This file can then load a data file containing log data in tabular ASCII format.

Output Formats Gnuplot usually outputs in postscript but it meanwhile also supports GIF output.

Plotting Style Support Gnuplot handles both curves (2 dimensions) and surfaces (3 dimensions). Surfaces can be plotted as a mesh fitting the specified function, floating in the 3-d coordinate space, or as a contour plot on the x-y plane. For 2-d plots, there are a number of plot styles, including lines, points, lines with points, error bars, and impulses (crude bar graphs). Graphs may be labeled with arbitrary labels and arrows, axes labels, a title, date and time, and a key.

However simple standard plots like bar chars, pie charts, filled boxes are not supported out of the box and only possible using third-party hacks and only with very limited functionality.

Statistical Capabilities Gnuplot supports all the standard functions of the standard UNIX math library therefore complex arithmetic is feasible.

Interactive Gnuplot can be used in interactive mode if desired. This is not helpful for usability evaluations however.

Ease of Learning Usability I spend quite a while trying to learn gnuplot to achieve the plotting results I wanted with little success. Only with applying the above mentioned hacks I got somewhat close to plotting specific bar charts but still lacking in many aspects. In my opinion the learning curve for gnuplot is very steep with low ease of learning usability.

Conclusion

Although the first approach with VizWear is nicely applicable in general settings, I see a number of problems.

No Real-Time Due to the nature of the audio-sync processing of video data, and probably other reasons, live, real-time visualization is completely lacking here. However I consider it important to be able to observe live visualizations in difficult usability tasks to quickly assess what is going on with the user performance in sometimes hard to overlook ubiquitous environments.

No Scripting This approach supports no scripting whatsoever. The tool will simply show all data it has. It is not possible to select just certain parts of the log data. With scripts the log file could be parsed much more easily for complex queries and much richer visualizations than just time durations or time mapping of discrete events.

Still, VizWear might make a nice addition to our system to add synchronized video & audio capture.

Ploticus seems well suited to make typical graphs in very high quality, provided the supplied support for statistical functions is sufficient. It is not a function plotting package and has little support for mathematical formulas or scientific notations.

I do think however that it is well suited for the visualization of usability data. As mentioned quick experiments with its scripting language have demonstrated how easy it is to use for rapid proto-typing of graphs.

Additionally it does have a newsgroup[2] which is very actively visited even by the developer. I actually was in contact with the developer through this medium.

Even if it turns out later that ploticus is lacking some functionality, an arbitrary filter like a statistical package can be easily used as a mediator for pre-processing of log files before piping them into ploticus as the final render engine.

Gnuplot is maybe a good tool to serve as a filter but it is way too cumbersome to produce simple graphs like bar charts typically required for usability evaluation visualizations which are somewhat more "business-style" than what gnuplot is meant to support.

So although it is theoretically much more powerful than e.g. ploticus in its mathematical capability, it definitely asks for much more effort to draw even simple graphs while the added functionality is not really required for our purposes anyway.

So basically the mere lack of usability of gnuplot compared to the ease of learning and use of ploticus convinced me to go for ploticus. I have not been familiar with either tool prior

[2] http://groups.yahoo.com/group/ploticus/

to this research. Spending time with ploticus yielded good results much faster than what I could gather with gnuplot scripts.

8.3 Action Visualization

Wearable

Wearable interaction poses new problems for usability evaluation studies. The great flexibility of these devices stem from their advantage to take the context of the user into account, when she wanders about with the worn device. In general it can be said that context-aware applications create new challenges in designing usability studies.

Summary In order to fully understand how a user interacts with a wearable, the researcher must examine both the user's direct interactions with the computer, as well as the external context the user perceives during the interaction. An attempt was made to develop a general, reusable tool to aid the researcher in studying wearable computer interactions by capturing the use of the machine in the field under normal circumstances [38].

Internal Structure To achieve this observation a mobile, multi camera system was used as shown in Figure 8.4 while trying to minimize the impact on the wearable computing experience. To do so an existing wearable was augmented instead of designing a completely new machine.

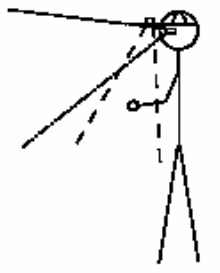


Figure 8.4: Camera Configuration for Wearable Action Visualization (Courtesy of Starner et al.)

To try to capture what the users sees and hears the system records video from cameras placed near the eye. One camera looks forward attached to the HMD and the other one, attached to the hat points down towards the user's hands to see e.g. how the interaction devices are handled. However the system is flexible and any number of cameras could be theoretically added to collect multiple video streams.

By using tapes to record data instead of Digital Video (DV) no performance impact was added to the wearable system but this of course resulted in more equipment to be worn by the participant. To store all this equipment a special capture vest (Figure 8.5) was used.

Figure 8.5: Capture vest (Courtesy of Starner et al.)

Additionally using a VGA to NTSC converter as a mediator between the wearable and the HMD, the output of the HMD itself was logged. This works fine as long as the resolution of HMDs does not exceed that of NTSC.

Conclusion

This approach seems reasonable to me but it has a number of problems. By placing cameras like that fix to the users body, fix assumptions have to be made where the user's hands are at all times. Of course when fixing the input device e.g. on the breast, pointing a camera exactly there will get all the interactions with this device on video.

But this approach will fail when the device is for example worn on the users hands like devices used in my sample usability study described in part IV. Some users will put their hands straight ahead, some will put the hand near the hip area and some will maybe put it high up in the air to control the device. I see two solutions for this problem which need further investigation.

Track Hands By tracking the user's hands, cameras attached to the body on flexible, motorized arms could automatically follow the movement of the hands and always capture them. However this form of construction will put a heavy, probably intolerable burden on the participant which will very severely impact the user experience. Still it would allow maximum mobility.

Place Cameras in Environment This approach does not suffer from these problems. Here the cameras are placed in the environment in strategic locations. Of course this limits

the mobility of the user. I can however envision setting up cameras in a certain city district for usability evaluations. Video processing technologies are already in development which can recognize certain faces so the cameras could be set up to record only the participant from all important angles. Lacking these technologies, it would be even easier to just place small tracking devices on the participant which are picked up by the cameras.

With this methodology the user experience is not harmed in any way by the usability evaluations plus this video data is probably much better suited to get the big picture. Just seeing the hands or the users view is not everything a usability monitor needs to see. Often it is also very interesting to observe the whole body posture or simple facial expressions.

9 System Design

I have designed a software solution for usability evaluations in Ubiquitous Computing using both DWARF and off-the-shelf products.

This chapter contains the design of the software part of the proposed usability evaluation framework, which is the core of my thesis. After presenting the design goals, I will divide the solution into subsystems to reduce complexity. Finally the implementation details of the identified subsystems are shared.

Action visualization is not handled because its implementation was out of scope for this thesis. Refer to Section 10.3 for implementation details of this component.

9.1 Design Goals

The most important design goals for this system were:

Good documentation: This diploma thesis may be the base for future projects.

Deployment Cost: Future projects will have new usability evaluation monitors who should be able to install and learn the system in a short time-frame.

Extensibility: The system should be extendable to support more elaborate usability evaluations in future setups.

Modifiability: New studies will require possibly new data log visualizations combined with new measurements to show deficiencies. Therefore the system must be changeable to accommodate this.

Utility: The system should support the work of a monitoring usability engineer during his tasks optimally.

Re-usability: The system should be re-usable on multiple levels that is written scripts for visualizations or the data logging components themselves should be designed appropriately.

The following section shows how the goals have been incorporated into the design.

9.2 Subsystem Decomposition

Please see figure 9.1 for an overview on the subsystem decomposition.

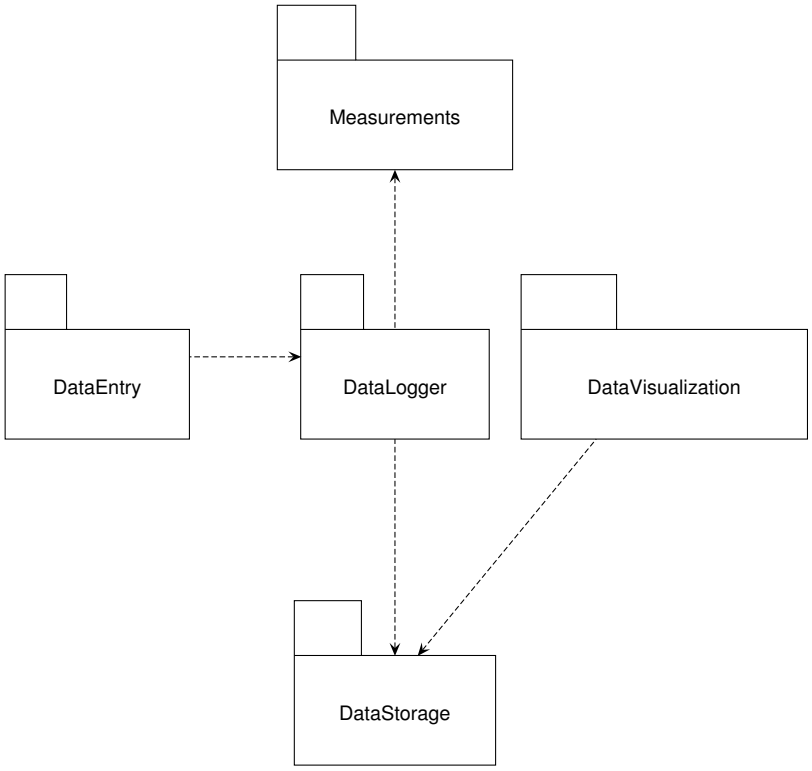


Figure 9.1: Subsystem Decomposition

Measurements This subsystem is made up of an arbitrary number of further systems which communicate with each other, thereby emitting data which is interesting for data logging.

DataEntry This component is used to manually enter data into the logging system. This includes notes but also manual task time taking. It is dependent on the DataLogger subsystem to propagate its manual data to.

DataLogger The main logging subsystem receives manual data from the DataEntry subsystem and logs the communication of services in the Measurements subsystem. It depends on the DataStorage subsystem for persistent log-file storage.

DataStorage All storage concerns are tackled in this subsystem. It only communicates with the DataLogger subsystem.

DataVisualization The remaining major subsystem depends on the DataStorage subsystem for data to analyze, interpret, and visualize.

The subsystems can be further split up into smaller components as in figure 9.2 for finer granularity.

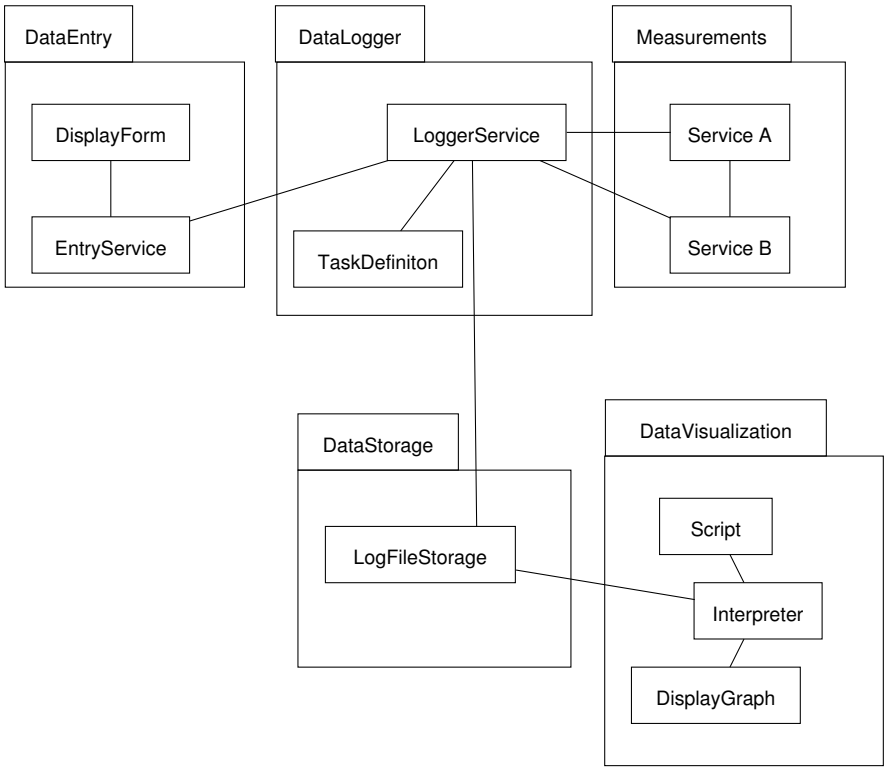


Figure 9.2: Detailed Subsystem Decomposition

Measurements Two or more services e.g. “ServiceA” and “ServiceB”, which can be of any type, are exchanging information in this subsystem.

DataEntry This subsystem can be split up into two. The “EntryService” actually communicates with the “LoggerService” in the DataLogger subsystem and asks for persistent storage of manually entered data. The “DisplayForm” component handles the display of an entry mask used to type in manual data.

DataLogger The “LoggerService” taps into the communication of “ServiceA” and “ServiceB”, and writes log data to “LogFileStorage” handled by the DataStorage subsystem. There is also communication with the “EntryService” to receive manual log data but also for control features such as setup of initial data or for receiving start and stop commands.

Finally a "TaskDefinition" component tells the "LoggerService" how to interpret the communication streams it tapped into, so it knows when a new task was started or a current one completed to handle automatic time taking and for incrementing the task counter.

To re-cap, a core messaging mechanism in DWARF is the dispatching of events which is based on CORBA [39]. The only other means for communication is the use of direct method calls from the sender to the receiver which are used rarely, and shared memory [37] for high bandwidth data like video-streams.

Services in DWARF have needs and abilities and register these with a service manager [39]. Events are send to an event bus and those services which registered their need for events of a certain type at the service manager will receive those events. See Section 3.3 for more details.

The DataLogger is meant to only log events. By doing so it is unobtrusive since this does not require any changes from other services. Services will exchange events interesting for measurement on their own, so it is just a matter of figuring out in which of these channels to tap into. Usually for usability studies in DWARF, just logging events should suffice, for my sample study it certainly has. It is technically easily possible to extend the DataLogger to also log method calls and shared memory activities in a future time if this is ever required though.

By leveraging this principle, the DataLogger effectively blends into the DWARF system and becomes a part of it, thereby supporting transparent logging in a distributed system. Additionally it doesn't take much effort to change needs in DWARF on a case by case basis. An edit of a XML file is sufficient. This aspect makes the solution very flexible too.

DataStorage By means of "LogFileStorage" persistent log file storage is assured, meeting an important functional requirement.

DataVisualization This subsystem can actually be split up into three components. The "Interpreter" takes interpretation instructions in form of a script from the "Script" component. The data which is to be interpreted is taken from the "LogFileStorage" component. Finally a "DisplayGraph" component is used to actually render the visualization instructed by the "Script" component.

So the interpreter has two inputs.

a. The log file produced from the DataLogger which is to be analyzed

b. A script which contains instructions on how to analyze and visualize the data

By splitting the input in these two parts, rapid prototyping is made feasible. If a new form of visualization is desired, merely a new script has to be put together. The interpreter should have all of the basic required functionality, so the usability monitor can focus on deciding on how to visualize the measurements and will not have to think about how to plot a bar chart.

9.3 Hardware/Software Mapping

See figure 9.3 for an overview on the system deployment.

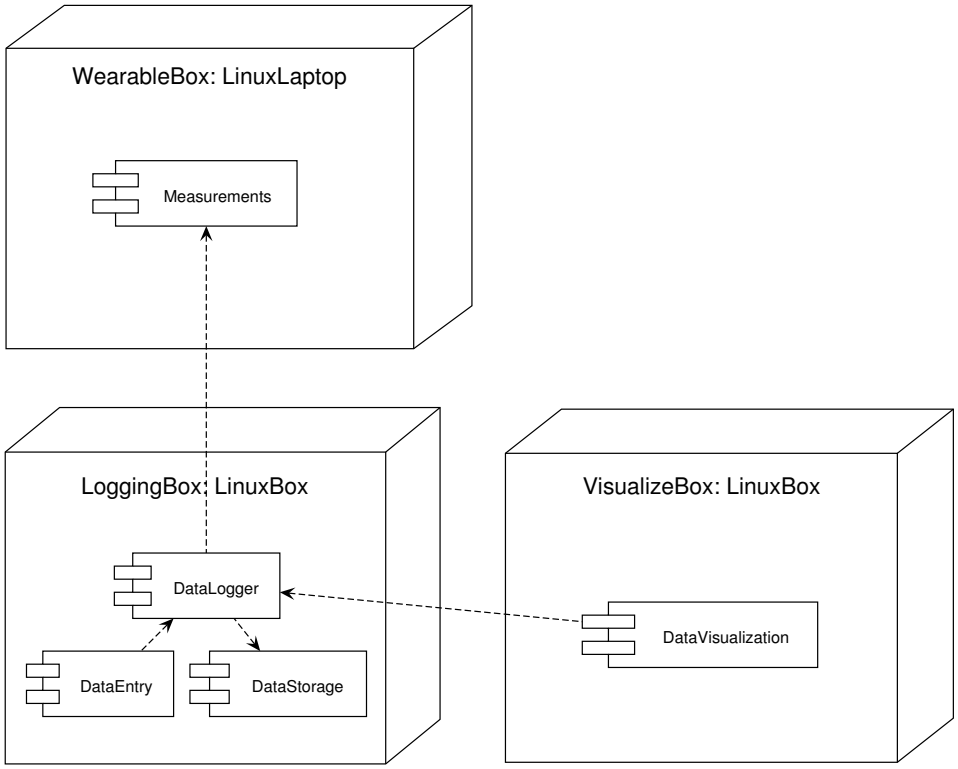


Figure 9.3: Deployment Diagram

At least three computers take part in one given usability study. One computer is worn by the participant outputting measurement data, one computer handles all the logging, and the third computer covers the data visualization.

WearableBox Some of the services of the user interface are running on the wearable rack worn by the mobile participant. To simplify, let's assume all services including the complete application run on this mobile setup. The rack has two mounted laptops for distributed processing to ensure smooth client operation.

Both laptops are ordinary Linux Intel-based PCs with 2.4GHz and 1.8GHz Pentium IV processors, each with 256MB of RAM. A VGA port is provided to attach an HMD. Both PCMCIA slots are filled with wireless LAN cards for intra-computer communication within the DWARF system.

Both laptops have one serial port to attach wearable input devices. One laptop has an IEEE 1394 interface for camera mounting to enable context awareness [37].

LoggingBox Here all the logging subsystems are running, including the entry mask and the log-file storage for easy administration by the usability monitor.

The logging computer is an Linux AMD-based PC with a 1700+ Mhz Athlon XP processor and 256MB of RAM. It is connected to the DWARF communication system [39] by means of a standard Ethernet cable connection.

This computer is part of a Network File System (NFS) where it stores the log-file in.

VisualizeBox Here only the visualization subsystem is running, containing all the scripts for interpretation of data measurements.

The visualization computer is an Linux Intel-based PC with a 2.4GHz Pentium IV processor and 512MB of RAM. Like the logging computer, it is connected to the DWARF communication system through a standard Ethernet cable connection and is part of the Network File System to read the log-file written by the logging computer. The X-11 Windows system[1] is leveraged to actually render plots on screen.

It is important that this computer has a high amount of CPU power and is totally separated from the remaining DWARF system. No services are running here.

This is because it can be very CPU intensive, when several scripts are to be interpreted at once and updated in real-time based on voluminous data measurements. If the interpreter was running on the machine where the logging systems are placed, it could actually slow down these processes.

This would have a negative impact on the usability monitor efficiency, throw off time measurement precision, and could even have an impact on the participant's user experience if not all client services run independently on his wearable rack.

9.4 Persistent Data Storage

Persistent data storage is a core functionality for the logging system. The log file must be persistently stored to allow later review and interpretation. Still it should be possible to access the data while it is being written for live visualization of participant performance during a usability study.

First we need to identify what types of data have to be stored for unambiguous logging data.

Date & time For all time dependent data mining it is mandatory to log the current time in a detailed fashion with each logged event.

Study Multiple studies are to be conducted so their names should be stored to differentiate between them since usually measurements between studies are not to be mixed.

Participant A study is conducted with a number of participants. To facilitate data mining based on individual participants and to enable intra-participant comparisons some form of identification for participants has to be stored.

[1] http://www.xfree86.org/

Task Each study is made up of a number of tasks to be performed by the participants. To support e.g. time taking of tasks task names must be stored as well.

Type Within tasks, different types of events are usually interesting for monitoring. This might be a note, a specific event (for example "MenuSelection"), or special types for task time taking ("StartTask"and "EndTask") Also to e.g. count the number of different event types, for example the total number of button clicks, this data should be saved.

Value Finally the to be logged event might contain additional information which is log worthy. For example if a note entered by DataEntry is to be persistently saved, the actual note is remembered in this value. The value could of course be a blob containing all the data contained in the event body for later data mining.

A decision has to be made if to use a file or a database for storage.

File A single event with the above six components is not very voluminous. However, many such events are logged. Ideally the logger should log every possible event to keep all options for data mining. Tracking services emit "PoseData" events about 30 times per second. Accumulating all send events of all services will potentially result in very voluminous data speaking against using a file for storage.

Speaking for a file solution, the information density of the log entries is rather low and only one writer but multiple readers are required. Additionally third-party tools for data analysis usually work on flat files only.

Database A relational database might be very beneficial allowing complex queries on the logged data while handling the large dataset gracefully.

I thought about placing the log events into a database [66] but decided to use a flat file in the end. The components written to interface with the database were written in the same time this thesis was, resulting in unresolvable time conflicts. Additionally the third-party tool I decided to use for visualization required a flat ASCII file for input.

Concerning the huge potential of database queries for data mining, using a database for log file storage should be considered in future work.

To meet the requirements of the third-party tool, the following BNF-Notation is specified for the log-file:

$< LogFile >$	::= $< EntryLine >^*$
$< EntryLine >$	::= $< DateTime >$,$< String >$,$< String >$,$< String >$,$< String >$,$< String >$
$< String >$	::= "`string`"
$< DateTime >$	::= $< Date >.< Time >$
$< Date >$	::= $< month >/< day >/< year >$
$< Time >$	::= $< hours >:< minutes >:< seconds >$

Table 9.1: BNF-notation for DataLogger log files

`String` is a standard string of arbitrary characters. $< Month >$, $< day >$, $< hours >$, $< minutes >$, and $< seconds >$ must always consist of two numbers whereas $< year >$ must always have four. $< Time >$ is in the 24 hour format.

Figure 9.4 shows one sample log file section with the corresponding mapping.

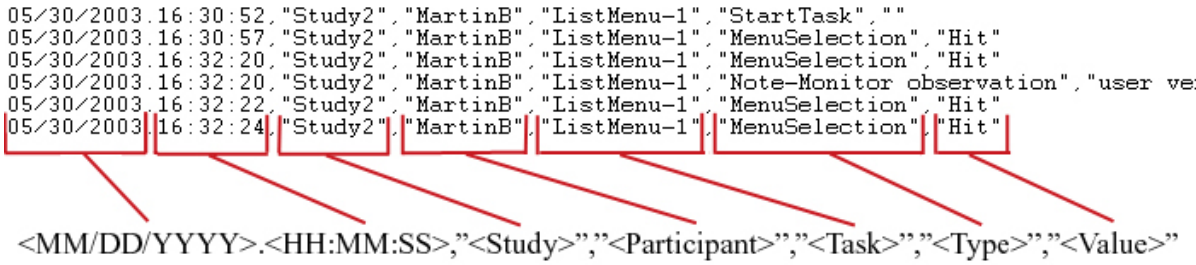


Figure 9.4: Sample log data file section

Each log file entry makes up one line in the file. Each entry must consist of the above identified six components, each separated by comma. Except the first component all components are enclosed in quotation marks to allow white-spaces.

9.5 Access Control and Security

Augmented Reality is still a field of new research, so up to now no security relevant applications have been developed within DWARF. In consequence, security issues and access control were no concern for the development of the logging system.

However, if strong security is to be implemented in the future, data logging in the current form will probably break. The idea of the un-obtrusive approach is to sniff into non encrypted communication between different services by registering a need for it. The current DWARF system will follow this request and forward all asked for events additionally to the DataLogger.

The DataLogger only works like intended if these events are readable to it. Theoretically the DataLogger could easily tap into the communication of a "Log-in Service" and an "Authenticate Service" to protocol e.g. passwords transmitted in clear-text between these two services used to authenticate users logging into DWARF.

DWARF is in fact inherently insecure as of now. Within ARCHIE, a service `DISTARB` [66] was written, which was used to transmit fake, dummy data. As long as DWARF allows arbitrary service connections, services such as `DISTARB` can be leveraged to break into the system by pretending to be any service to log any data or to trigger any desired reaction.

It is future work on how to handle usability data logging when access control is added to DWARF.

9.6 Global Software Control

The visualization tools are widely-self-contained. A monitor thread is used to poll the log file in regular intervals to check for changes to know when to update the visualizations.

Both the DataLogger and DataEntry are separated services running in their own threads.

9.7 Boundary Conditions

In this section boundary conditions which can occur are described.

Initialization The visualization interpreter is initialized by supplying the script-file and the log-file as command line parameters. Please refer to the Appendix A.1 for further details. The `DataLogger` can be initialized in form of the use case already listed in Section 7.2.

Failure of components This subject is beyond the scope of this thesis. Further study is required for these points:

- Visualization errors due to missing matching log data
- Visualization error due to reaching a process limit of the interpreter
- Network connection issues

10 Implementation

The first implementation covers all the functionality to begin conducting usability evaluations of DWARF applications.

This chapter provides insights on the current implementation details of the software framework for usability evaluations in DWARF.

See figure 10.1 for a quick overview about all major components.

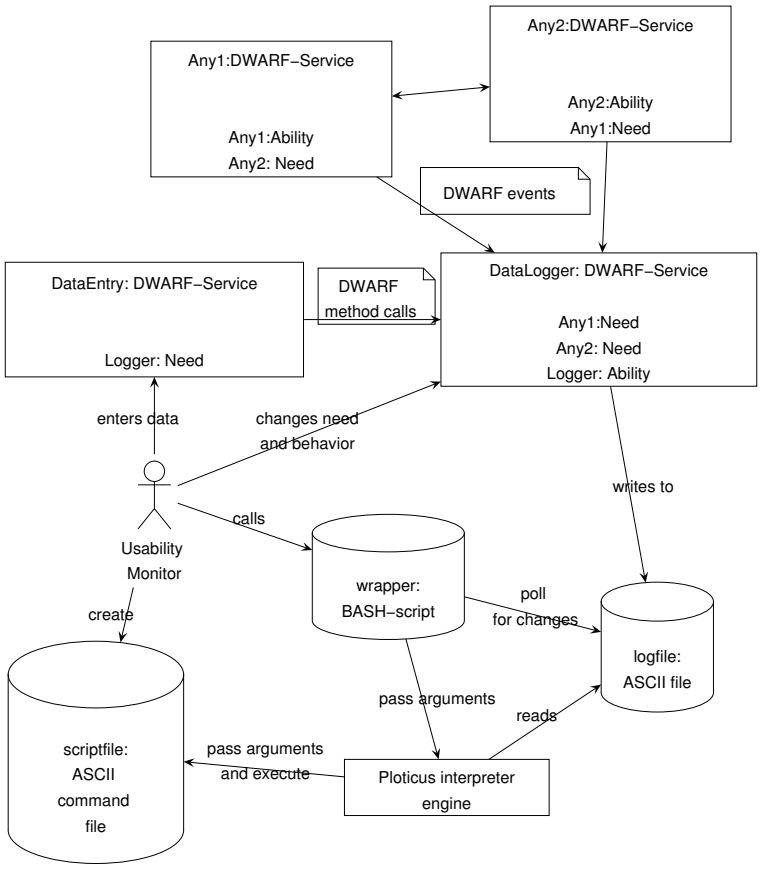


Figure 10.1: Implementation Overview

`DataLogger` This DWARF-Service has the need for various types of events as measurement data. These needs are changed on demand by the usability monitor as is the internal behavior on how to handle each type of event. The current implementation does not tap into every event channel but just those which have been specified as needs. Besides being able to receive events, a method call interface is offered to `DataEntry`.

In the above overview `Any1` and `Any2` exchange events by mutual ability and need relationships. `DataLogger` listens to both event channel directions by registering a need for both types at the service manager [39].

Data is written to a standard ASCII file by appending.

`DataEntry` Based on QT, to display a simple UI for manual data taking, this DWARF-Service has merely the need for the method call interface made available by `DataLogger`, to propagate data to.

Wrapper bash script Based on the adapter design pattern, these scripts are called by the evaluation monitor to request visualizations. They offer a common interface to all custom scripts, and handle all the parameter passing while polling the logfile, written by the `DataLogger`, regularly for updates. Refer to Appendix A.1 for usage guidelines regarding this wrapper.

Ploticus Interpreter This is an already existing third party tool for interpreting and plotting data based on scripts.

Scriptfile These are a collection of scripts for the ploticus interpreter, written by the usability monitor. Refer to Appendix A.1 for details on the scripts which were written for this thesis.

Logfile This file is written by the `DataLogger` and read by arbitrarily many ploticus interpreter instances.

Going into more detail `DataLogger` followed by `DataEntry` is described. Finally the data visualization implementation is discussed.

10.1 `Data Logging`

DIVE (DWARF Interactive Visualization Environment) [47] visualizes the interactions between different services. To see how the `DataLogger` operates in the running DWARF system, a snapshot of one sample system state is shown in figure 10.2.

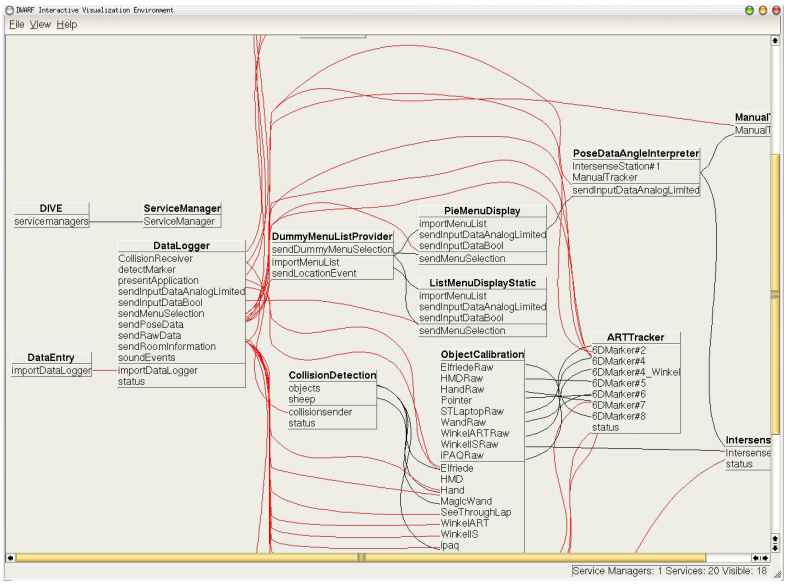


Figure 10.2: DIVE DataLogger integration

We are only interested in `DataLogger` and `DataEntry` so everything else should be disregarded. However the graphic demonstrates how easy it is for the `DataLogger` to tap into many event channels at once. Every red line going out from `DataLogger` is one opened event channel to capture measurement data.

`DataLogger` The actual `DataLogger` service offers the "importDataLogger" ability and has many needs e.g. for "CollisionReceiver", "detectMarker", "presentApplication", "sendInputDataAnalogLimited", "sendInputDataBool", "sendMenuSelection", "sendPoseData","sendRawData", "sendRoomInformation", and "soundEvents" as depicted in the above system snapshot.

`DataEntry` The method call interface "importDataLogger" is exploited by `DataEntry`. By using method calls, we can be certain that no notes get lost due to possible, but unusual DWARF system lag issues.

Summary This service can receive a multitude of events (easily extendible) and log them to a file persistently.

Needs & Abilities

Ability DataLogger (SvcProtObjrefExporter)

Need MenuSelection (SvcProtPushConsumer)

The only ability offered by this service is an interface to be called by *DataEntry* for receiving manual entries.

The *DataLogger* service can have an arbitrary amount of needs for different types of events. In this example one need "MenuSelection" is listed because this is the only one which was required for the sample study I conducted in part IV. The abilities will not change from study to study but the needs will have to be adapted accordingly (by editing the XML service description), after it has become clear which data should be measured / logged for later analysis.

Internal Structure The `DataLogger` receives events and extracts information interesting for logging and writes each such occurrence out as one line to a log file in the previously described BNF-format.

The *Date & time* value is generated by the `DataLogger` itself each time it writes a log file entry. *Study*, *Participant*, and *Task* are initially set to "?" since the `DataLogger` has no way of knowing which user is currently using the system, or which study is currently conducted. The initial task name is also unknown at first. These values are set with the complementary tool *DataEntry* described in the next section. DWARF is being constantly extended though so there is a good change some of these values like currently logged in user can be automatically extracted in the future.

Type refers to the type of event which was logged and *Value* is usually some value extracted from this event. For example if the event was of type "InputDataAnalogLimited", *value* would usually carry the actual number. This can be easily changed on demand however depending on the concrete requirements for the study.

The user may specify the desired log file and enable an automated task counter with command line parameters.

-Dfile The full path of the file to be logged to (new lines will be added to the bottom)

-Dcounter Any value enables the task counter.

Initially the task counter is set to 1. If the counter is enabled, the `DataLogger` will always put $< TaskName >$-$< CounterValue >$ in the *Task* field when writing events. Often the logger knows when a task is completed, so it can increment the task counter automatically. This is very useful when the participant of an usability evaluation is asked to perform the exact same task multiple times to observe learn effects. The `DataLogger` also has the capability to take time automatically if it's possible to interpret received events in such a way to find the task start and end events.

All of this was leveraged in the sample study conducted during the course of this thesis which allowed me as the usability evaluation monitor to focus my attention on the actual participant since I was not required to hit buttons during the study at all, unless I wanted to take notes with `DataEntry`.

See figure 10.3 for an UML class diagram on `DataLogger`.

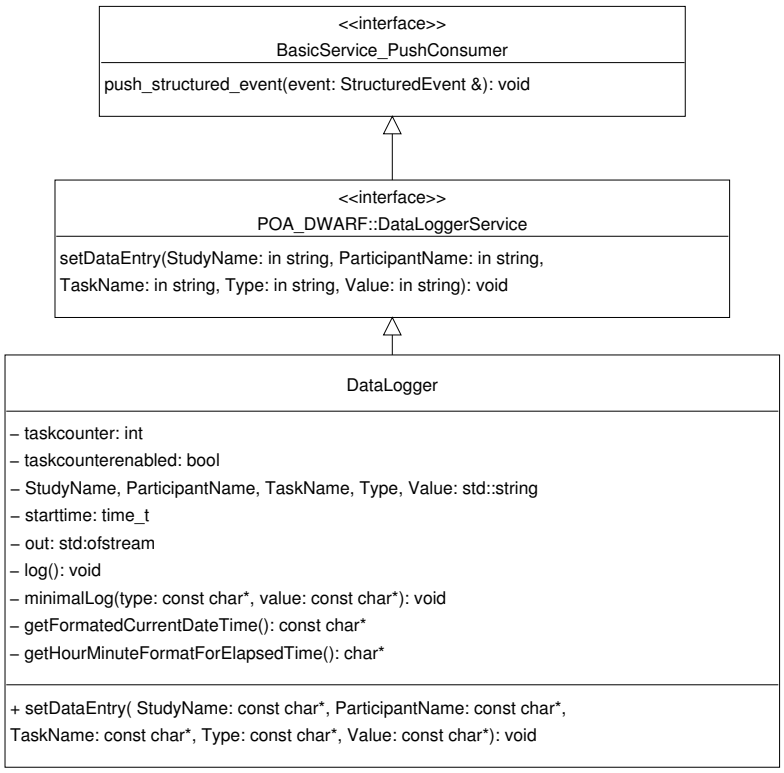


Figure 10.3: DataLogger UML Class Diagram

`DataLogger` implements the `DataLoggerService` interface which adds the method `setDataEntry()` for manual data entries to the `BasicService_PushConsumer` interface mandating `push_structured_event()`, which is always called within `DataLogger` when any event it listens for has "arrived". The current implementation will extract header information and then branch on a case-by-case basis and log different data depending on the extracted header information.

The internal variables *StudyName, ParticipantName, TaskName, Type, and Value* are written to the file *out* with a call to the private method `log()`. Usually however only `minimalLog()` is used to just specify a temporary *Type* and *Value* which is then combined with the current time and with the already stored *StudyName, Participant* and *Task* variable contents.

Whenever `setDataEntry` is called every passed value overwrites local variables. This is leveraged by `DataEntry` to pass the initially to "?"set values of *StudyName*, *ParticipantName* and *TaskName*. Unless both *Value* and *Type* are set to "INIT-VALUE"the passed variables will be immediately written out to the logfile.

The internal method `getFormatedCurrentDateTime()` outputs a string which is in accordance to the earlier specified BNF-specification. When *taskcounterenabled* is set to "true", by the above mentioned command line parameter passing, automatic logging is "enabled". In these cases the value of *taskcounter* is added behind the *TaskName* for each single log entry. The taskcounter is usually incremented within the above mentioned case branch in the

`push_structured_event()` method. Here also *starttime* is set and compared with a later system time to measure elapsed time which is converted to a ploticus usable time format with an intermediate call to `getHourMinuteFormatForElapsedTime()`.

So basically the only way to define tasks as of now is by hard-coding all the logic within the `push_structured_event()` method. For repeated tasks it is sufficient to increment the *taskcounter* appropriately. Otherwise the content of the local variable *TaskName* itself would have to be overwritten with e.g. contents taken out of a newly received "Control-Event". For ploticus time range visualization a log entry with both *Type* "EndTask-Success" and *Value* in the `getHourMinuteFormatForElapsedTime` must be written. The ploticus scripts rely on this set *Value* field, although they should compute the passed time automatically, by comparing the first log component *Date & Time* of the events with matching *Study, Participant, and Task* and with "Starttask" and respectively "EndTask-Success" for *Type*. Implementing this is future work.

There currently are no means to stop the `DataLogger` in between operation. Additionally no feedback channel back to `DataEntry` was yet implemented.

Manual Data Entry

Summary This complementary tool based on QT [18] enables the evaluation monitor to initialize the *DataLogger* with the correct *Study*, *Participant*, and *Task* values, note task start and end times manually, and enter comments which are saved persistently in the same log file as described above.

Needs & Abilities

Need DataLogger (SvcProtObjrefImporter)

This service has no abilities to offer to other services. It merely requires the *DataLogger* to be able to initialize it or to propagate log events to it.

Internal Structure The following figure 10.4 shows the *DataEntry* mask in two states.

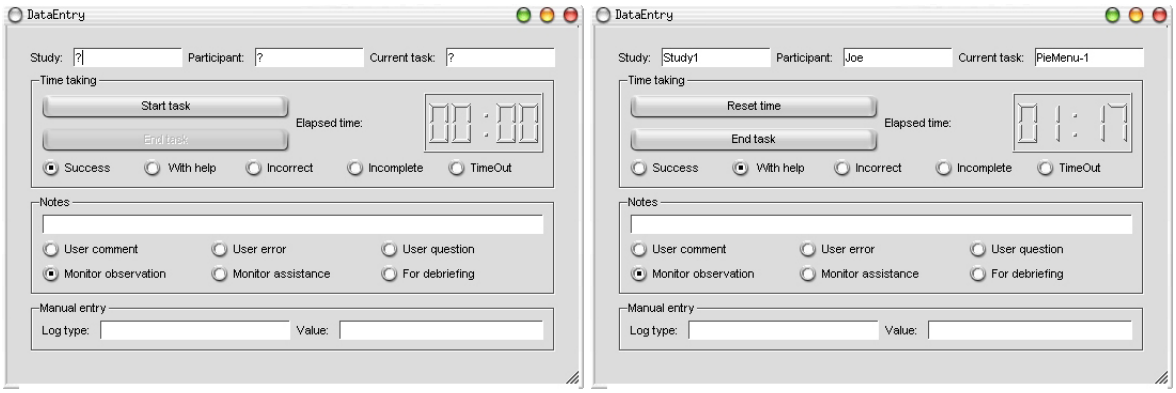


Figure 10.4: DataEntry mask, Left: start; Right: init values entered plus timer started

The left state is the initial one which is shown when it is first started. Like mentioned in the *DataLogger* description, the values for *Study*, *Participant*, and *Task* are initially set to "?". This mask is meant to be used to set the correct values. For this the field just has to be selected by clicking or tabbing into it. Once the correct values have been entered, hitting "Enter" is sufficient for confirmation and for setting up the *DataLogger*.

Times can be manually taken by pressing the `Start task` button and then pressing the `End task` button after making sure the proper radio button (Success, With help, Incorrect, Incomplete, Timeout) is selected. For `Start task` events, *Type* "StartTask" is logged with no *Value*. In case of `End task` events *Type* "EndTask-$Radio - Button - Selection$" is logged with the time shown on the updating LCD as *Value*. For example if the `End task` button was hit in the mask to the right in figure 10.4 *Type* "EndTask-With Help" would be propagated together with *Value* "01:17" to the *DataLogger*.

Once the `Start task` button was hit, it automatically changes into `Reset Time`, which when clicked merely resets the time with no propagated log event. This is useful if the `Start task` button was accidently hit too early or if the participant had to restart his task due to technical issues. The `End Task` button is disabled when no time is currently "running" like shown in figure 10.4 on the left.

Notes can be taken by entering the desired text in the `Notes` field and pressing enter. The note type will be set to the radio button (User comment, User error, User question, Monitor observation, Monitor assistance, For debriefing) checked below the notes field. The *Value* is set to the actual note whereas *Type* is set to "Note-$Radio - Button - Selection$". If a note was propagated of type "Monitor assistance", the radio button "With Help" is automatically selected in the time taking form.

Manual entries where both *Type* and *Value* can be freely chosen can be send to the *DataLogger* by using the bottom most form by specifying the `Log type` and `Value` each again confirmed by pressing "Enter".

See figure 10.5 for an UML class diagram on `DataEntry`.

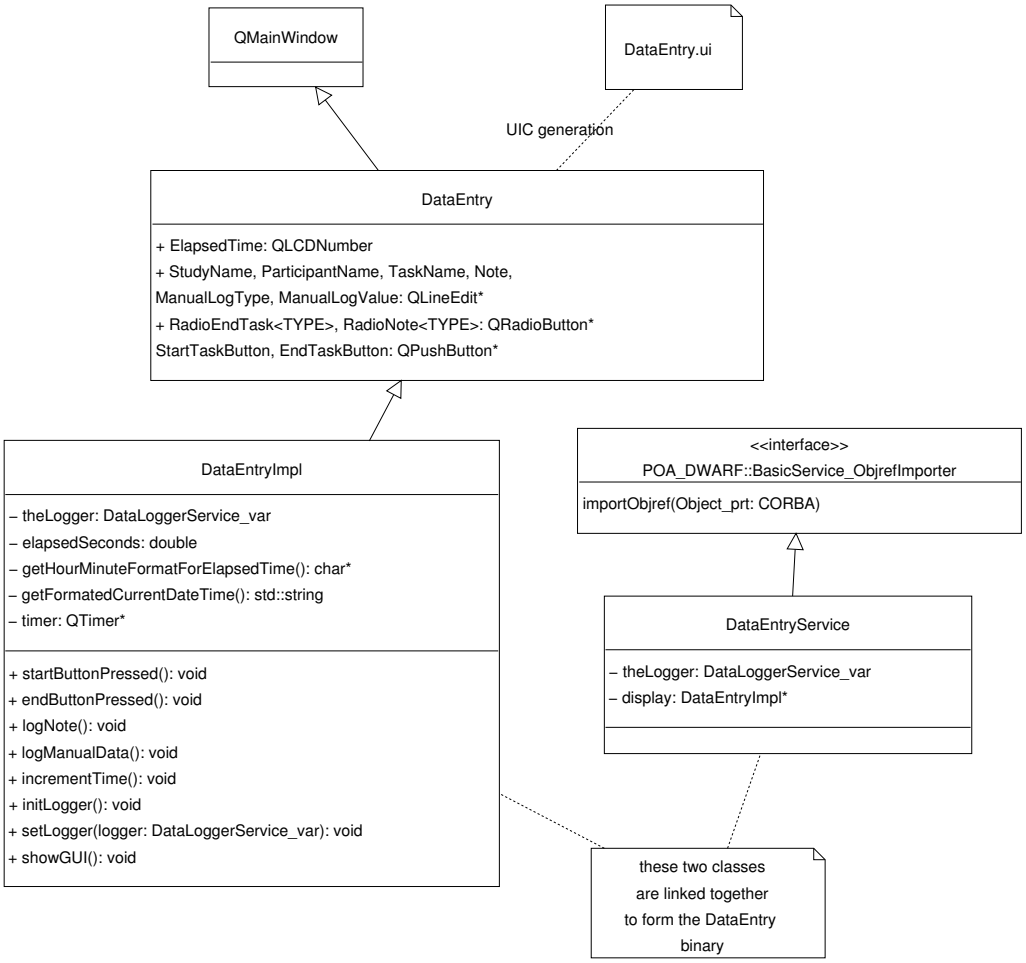


Figure 10.5: DataEntry UML Class Diagram

`DataEntry` is linked together from two different classes `DataEntryImpl` and `DataEntryService`.

`DataEntryService` This class implements the `POA_DWARF::BasicSerivce_ObjrefImporter` interface which forces the `importObjref()` method used to connect to `DataLogger` for data propagation via method calls. This class sets the variable *theLogger* and passes it on to its *display* which is made up of the second class.

`DataEntryImpl` DataEntry.ui was generated with QT designer for high level user interface design. Buttons, radio-buttons, text-fields and one LCD has been added to the layout whereas all the components received meaningful names.

UIC, a generator tool provided by QT designer, generates a matching `DataEntry` class which is implementing the `QMainWindow` interface. This class has all the above mentioned public member variables for easy access by the main class `DataEntryImpl`.

The QSlot mechanism was leveraged to trigger the method calls `startButtonPressed()`, `endButtonPressed()`, `logNote()`, `logManualData()`, `incrementTime()`, and `initLogger()`.

Whenever any data is propagated to the DataLogger, StudyName, ParticipantName and TaskName are taken anew from the mask to be send via method call.

initLogger() This method is called when the usability monitor hits Enter in either of the QLineEdit fields StudyName, ParticipantName or TaskName. "INITLOGGER" is used for both *Type* and *Value* when calling `setEntry()` on the logger.

logNote() The current value of the Note QLineEdit field is used as *Value* and for *Type* a combination of "Note-" with the selected radio button is propagated.

logManualData() The current values of the ManualLogType and ManualLogValue QLineEdit fields are used as *Value* and *Type* for propagation to *theLogger*.

startButtonPressed() When this button is pressed data with *Type* "Starttask" is propagated with no value, while starting the *timer* which is based on QTimer. After doing so, the end button is enabled and the start button changes into a reset button to reset the *timer*.

incrementTime() When the *timer* was started by a previous start button press, this slot is called every second to update the *ElapsedTime* QLCDNumber.

endButtonPressed() `SetEntry()` is called on *theLogger* with the *elapsedseconds* as *Value* after they have been converted to the correct format with an intermediate call to `getHourMinuteFormatForElapsedTime()`.

10.2 Data Visualization

I decided to use "ploticus" for the plotting interpreter engine because of the features listed in Chapter 8. It met the requirements pretty closely out of the box:

Rapid Prototyping By using a scripting language, new visualization scripts can be quickly put together.

Statistical Functions Ploticus comes with a number of default statistical functions sufficient for our purposes at the moment. If they do not suffice later on, ploticus could be extended by editing the source code (it is licensed under the GPL). The libploticus C language API could also be leveraged directly. Otherwise the logfile could be pre-processed by another tool like SPSS[1], should it be required in the future.

2D Plotting Ploticus offers only 2D plotting which is fine with our requirements. It supports many different plotting styles like bar chars, scatter plots, range bars, pie charts out of the box which again facilitates rapid prototyping of scripts.

Unobtrusive/Transparent Ploticus can run on a number of different platforms totally independent from the remaining logging system.

[1] http://www.spss.com/

Off-line High Quality Ploticus offers a variety of output options, including postscript formats which support free-type and true type fonts for high quality renderings.

Real-Time This was the only requirement lacking from ploticus but a solution was found based on shell scripts which added an real-time display as described in the Appendix A.1. Basically these scripts just run in the background and check in regular intervals, by polling, if the log file has changed. If it has, the current view is terminated and a new one is rendered by calling the interpreter again.

Please refer to the Appendix A.1 for detailed usage descriptions scripts already written for this thesis.

I recommend going through the already existing scripts since they have very detailed inline comments and to read the excellent handbook available at [4] to get started with compiling new visualizations.

In all cases the provided `pl_wrapper` script should be used for convenient parameter passing and to leverage the real-time features.

10.3 Action Visualization

A solution based on video-see-through was written by another ARCHIE team member [30]. Here a small camera is tracked which can be placed anywhere freely. The visual augmentation is then registered and overlayed on this camera picture in real-time. The camera was using a Fire-Wire interface and could be easily connected to a laptop to fulfill the mobility requirement. Persistence could have been achieved by writing the shared memory buffer to disk constantly but due to the reasons mentioned in the requirements elicitation in Section 7.3 this was not attempted. For the sample study I conducted, an even simpler solution was used as described in part IV.

10.4 Restrictions of the Prototype

Although the prototype already has a high level of utility to assist an usability engineer in his monitoring task, there is still lots of future work to be done especially if it is the aim to move from quick explorative usability evaluations to more elaborate ones.

Data Logging The logging process itself has potential for future growth in two aspects.

Semantic mapping Right now handling of events received by the *DataLogger* is hard-coded in the logger itself. By extracting the header of received events, the code branches and decides on a case by case basis which data to extract from each event type and how to log it.

To speed up the prototype development process this type of semantic overloading of the *DataLogger* was chosen. For example by analyzing the name of selected menu entries it could recognize if a “Hit” or a “Miss” was performed. This fact was exploited in the sample study described in part IV.

Ideally this would not be required. The logger would just tap into every possible channel, and log all data contained in every event in a blob-like format which could then be read by the visualization tool.

This would of course result in enormous data log volumes though and exacerbate the visualization of logged measurements with the proposed visualization solution for two reasons. First the visualization tool is slowed down if it has to process larger log files by itself. Second by adding no semantics to the logged data, all of this will have to be added to the visualization tool in form of e.g. more elaborate scripts which would take clearly more effort due to missing object orientation in scripts.

For rapid prototyping it has become clear to me that the chosen solution was superior although it is not ideal for the long run.

User Interface Controller The current prototype supports no logging of the User Interface Controller (UIC). In DWARF this component is based on the Petri Net framework Jfern [19] to model multi-modal interaction which is common practice in the area of work-flow systems [6]. Figure 10.6 shows one such example Petri Net. Tokens are put into places in the Petri Net. Only when all places on incoming arcs of a transition are full, a transition is triggered [41].

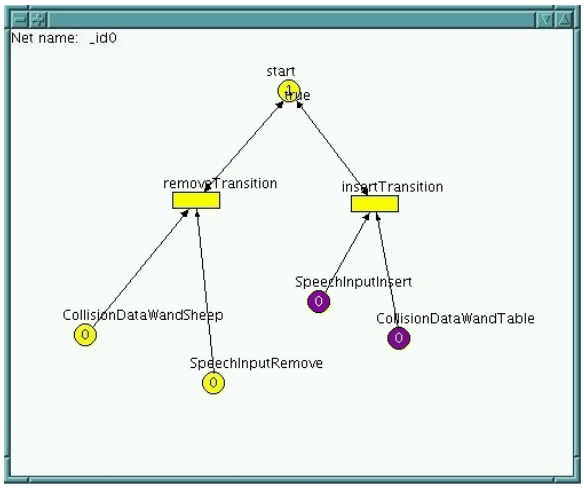


Figure 10.6: Jfern example Petri Net

This UIC would be a good place to model usability tasks which are to be performed by the usability evaluation participants. The *DataLogger* could of course also exploit this added functionality. Plus Jfern does offer another free visualization helpful for the monitor.

In the long run I think it is advisable to use the UIC to put usability tasks together for easy later re-production without relying on special components like how it was done in the sample study.

Data Entry The *DataEntry* tool also has potential for future work three-fold:

Mobile Solution The form is fine as long as studies on Ubiquitous Computing are conducted in the controlled environment of a laboratory. However I think studies really start to get interesting if they are carried out in their actual intended environment outside in the "real world". Here the monitor will be mobile too and will need a very different form of interface to the *DataEntry* tool which is feasible with wearable input devices.

Integration The integration with the *DataLogger* can still be improved. Currently time taking is either done manually using *DataEntry* or automatically with the *DataLogger*. If the *DataLogger* is taking times it will not trigger the time LCD on *DataEntry* which should be added in future work. This can be easily accomplished by adding a second communication channel between *DataEntry* and *DataLogger* but this time in the opposite way.

Time Out "Time out" is one of the radio buttons selectable in the time taking form. It is meant to be selected by the monitor if the participant has exceeded the maximal alloted time for the task. In the long run e.g. the UIC should be used to model tasks, and there this maximal time should be encoded as well to remove the need to click on this option. The same argument is valid for some of the other options.

Data Visualization Although the taken approach is perfect for rapid prototyping, it does lack severely in extensibility. In the long run it should be replaced by an object-oriented software solution.

With the prototype the monitor has to know in advance what he wants to visualize and prepare scripts taking this into account. During the study he will then start these scripts manually or activate the real-time updating visualization which will show a number of visualizations chosen fix earlier.

It is future work to make this approach much more flexible than it is now. I think great benefit could be achieved by combining the data visualization effort with the DIVE [47] system introduced earlier. For example by clicking on communication lines between services in DIVE, the visualization environment could guess wanted visualizations and render them.

Of course it would also be possible to integrate the *DataLogger* in a way like this but I don't think this is a good idea. Sometimes it gets apparent later in the study that a much better visualization can be achieved by combining measurements in way which was not considered before. Only when all data is logged in advance this can be done retroactively to all study results.

This was one of the lessons I learned while conducting the sample study which we will cover next.

Part IV

Sample Usability Evaluation

11 Sample Usability Study

A sample usability study was conducted to evaluate the proposed framework. In this chapter our options, implementation details and the actual study results are to be found.

To put the usability evaluation framework to a test and to collect practical experience it was decided to conduct a real, albeit small usability study.

It was not intended to conduct a thorough study yielding representative results but rather to test the applicability of the usability evaluation framework.

We still did mean to conduct a study in the Augmented Reality field with a topic worth pursuing. Searching for this topic multiple options were evaluated.

This chapter will give a rationale for the decision we have made after detailing these in section 11.1. Following the description of the decisions implementation (Section 11.2), the actual study results are presented in section 11.3.

11.1 Options

We started by looking for holes in the design space of all human computer interactions in ARCHIE. The design team was confident about most interactions of ARCHIE, but one was clearly the most troublesome.

Interfaces for system control tasks in virtual environments have not been extensively studied [14]. ARCHIE also had several such control tasks and needed some form of menu system to realize them. The implementation of this menu can be done in many different ways.

The remainder of this section will first list requirements for our implementation (Section 11.1.1) and then move on to list I/O hardware (Section 11.1.2). Combined with the design space for menu systems (Section 11.1.3) enough information has been collected to come to a decision (Section 11.1.4).

11.1.1 Requirements

Mobility On the user side the menu should be ubiquitous and thus applicable not only in stationary but also mobile scenarios for both indoor and outdoor support. A feasible mobile solution should also only require one hand for operation.

Layout On the author side the menu should feature automatic layout. Also referred to as "space management" here an attempt is being made of using the limited display space effectively, e.g. avoiding overlapping windows by re-shuffling them automatically in an appropriate manner [11]. For the study this aspect had only minor importance however, since we wanted to focus on the user experience.

Flexibility The solution should be flexible, meaning it should be versatile enough to support all menu scenarios.

Support for about seven entries The well known "Rule of 7" says that never more than seven elements should be displayed at once to not over-strain human brain capacity. [15]. Despite this we also noticed that we don't need more than about five menu entries in ARCHIE at any given moment of time anyway.

Operability To navigate the menu two operations have to be executable by the user. First an entry has to be selectable. The selection could take place via continuous or discrete input. Second confirmation of the previously selected entry must be possible via discrete input.

To understand these terms an input device classification (Figure 11.1) is helpful.

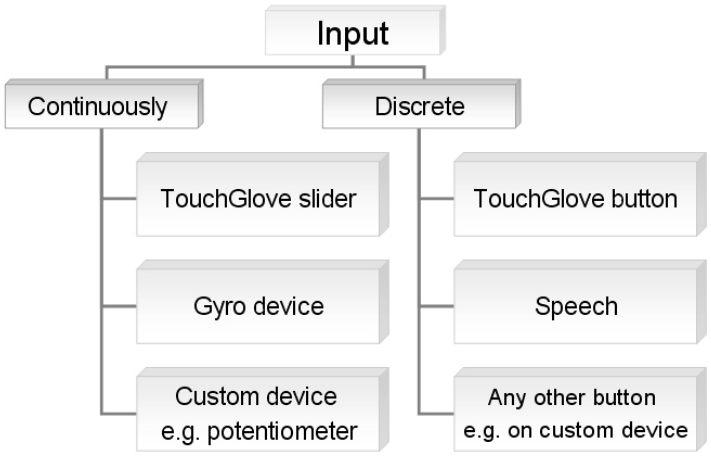


Figure 11.1: Input device classification

Input devices can generally be split into two categories. Continuous devices output data using a continuous data stream like e.g. any slider would, or gyroscopic devices like the inertial trackers. Discrete devices are typically buttons but also speech input. A single recognized speech token represents one discrete event.

Before discussing possible menu layouts, the available input and output hardware is shown in an overview. Subsequently hardware which we considered for purchase is also discussed.

11.1.2 I/O Hardware

It is important to understand which input / output hardware was taken into consideration to be able to follow the argumentation on possible menu systems. We will first tackle input and then output hardware.

Input Hardware

Certain hardware was already available to us by prior purchase. This is what the following section is about.

Devices Overview

ARTtrack1 This optical tracking system by the advanced real-time tracking GmbH[1] offers precise tracking for tangible input devices with attached optical markers provided there is no occlusion but its setup is inherently stationary. This system was available to us in a four camera setup. This device was used in prior projects, like SHEEP [41] to realize tangible user interfaces.

GPS tracking GPS tracking only makes sense in an outdoor setting.

Chameleon Touch-Pad The Chameleon Touch-Pad input device is a new input device developed by the Columbia University [13]. It consists of a half-glove and touch-sensitive surface device which is mounted on the palm portion of the half-glove. The physical design and the interaction method make it possible for the user to interact simultaneously with a wearable computer, as well as with objects and machines in the environment. Input is performed using both tapping and dragging motions of the fingertips on the surface (Figure 11.2, [76]). The device is connected to the computer via a standard serial port. The Chameleon Touch-Pad has the capability of being configurable to support both button and slider like behavior.

The picture shows how the Columbia University used this setup. We improved upon this as described in section 11.2.1.

Microphone / speech Hardware for Text-to-Speech synthesis (TTS) and speech recognition was available to us.

[1] http://www.ar-tracking.de/

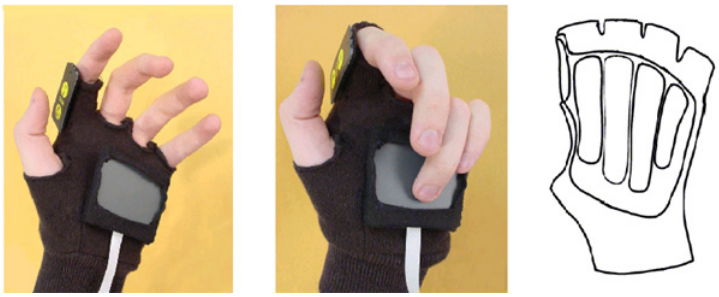


Figure 11.2: Chameleon Touch-Pad

Intersense InterTrax² This compact motion tracker by Intersense[2] (Figure 11.3) was designed as a head tracker attached to a head-mounted display to register e.g. the current point of view of the user by offering 3-DOF (degrees of freedom) angular tracking. It plugs into a standard serial port. The device is suffering from drifting issues after extended usage.

Intersense InertiaCube² Intersense also build the InertiaCube[2] (Figure 11.3) which again uses a standard serial port but offers higher precision at a full 360° range with multiple advanced features such as adjustable rotational sensitivity and motion prediction. Due to its higher update rate, tracker induced lag could be eliminated in this device. However it proves no immunity against drifting after extended usage either.

Figure 11.3: Left: Intersense InterTrax[2]; Right: Intersense InertiaCube[2]

Xsens MT9 The MT9 by Xsens[3] is yet another compact motion tracked cube which plugs into a standard serial port but with a major advantage compared to the Intersense

[2]http://www.isense.com
[3]http://www.xsens.com

InertiaCube^2. This much more sophisticated device has the highest level of precision without any drifting due to the absolute orientation reference provided by the accelerometers and the earth-magnetic field sensors added to the cube.

Custom input device with potentiometer and button The Carnegie Mellon University has build a dial device (Figure 11.4) which is a new, yet unpublished variation of the VuMan [61] product group for menu control. These devices use a rotary dial in combination with buttons. By arranging menu options logically in a circular list, a mapping from the one-dimensional input device to a two dimensional selection surface was achieved. Unfortunately none of these products are commercially available, but it has inspired us to think of a custom device with a potentiometer for rotational input and an attached button (Figure 11.4).

Figure 11.4: Left: CMU VuMan dial device; Right: mockup of custom potentiometer device

Discuss Devices

All devices but the Xsens MT9, the Intersense InertiaCube^2 and the custom device have been available to us already. We have experienced drifting issues with the Intersense InterTrax^2 earlier, so we considered purchasing either the Xsens MT9 or the Intersense InertiaCube^2 to eliminate these problems. Although the Xsens MT9 was superior to the Intersense InertiaCube^2, it would require completely new software on our end, since it is not compatible with our existing Intersense data interpreting software. I also saw great potential in the proposed custom device since it offered an intuitive, wearable interface while still in well defined levels of constraints to allow non tedious usage. Additionally this dial could be attached anywhere making it much more flexible than a rotation solution based on an inertial tracker.

For a quick overview all the above has been summed up in table 11.1.

This concludes the discussion of our input hardware, so we will tackle output hardware now.

Type	Cost for Us	Indoor	Outdoor
ARTtrack1	-	Yes	No
GPS tracking	-	No	Yes
Chameleon Touch-Pad	-	Yes	Yes
Microphone / speech	-	Yes	Yes
Intersense InterTrax^2	-	Yes	Yes
Intersense InertiaCube^2	aprox. $1.800	Yes	Yes
Xsens MT9	aprox. $2000	Yes	Yes
Custom device with potentiometer and button	aprox. $100	Yes	Yes

Table 11.1: Input Hardware Overview

Output Hardware

First actual hardware devices are listed, followed by modes which can be leveraged with them.

Devices

Mono beamer Two different beamer devices with supported maximal resolutions of 1024x768 and 640x480 primarily meant for workspace and video-see-through wall projections were available.

Sony Glasstron Model PLM-S700E Our first standard 2D HMD offered a resolution of 800x600.

Virtual I/O i-glasses! PC Version This HMD by iO Display Systems, Inc.[4] offers a line interleaved stereo mode but only up to an effective resolution of 320x480.

Since we wanted to obtain the highest possible mobile menu readability we opted for the Sony Glasstron 2D HMD as our menu display device.

We had the option of running multiple different modes on the above mentioned devices. This is discussed next.

Modes

Standard 2D All devices could of course be used in a standard 2D mode.

Anaglyphic stereo Merely special glasses are to be combined with any other output device provided the proper software is installed to pre-process the output device' display to use this mode. A sub-category of anaglyphic stereo is Red/Green stereo which suffers from very dark colors. Red/Blue or Red/Cyan combinations are superior in this aspect.

[4] http://www.i-glassesstore.com/

Video-See-Through Finally a video-see-through mode was being written with which it was possible to augment a video picture with virtual data properly registered with the image by using a tracked camera [30].

During the time of the search for a usability study objective, our main 3D viewer component was re-written from scratch [30] because the older viewer was inflexible, slow and incompatible with the remaining system unless additional wrappers were installed, which slowed the system down even further. Additionally both the new viewer and the sample usability study shared the same deadline. So it was decided not to wait for the completion of this new 3D viewer but instead rely on a standard 2D view for the sample usability study.

After settling on the used hardware and modes, the design space for menu systems has to be discussed to form a final conclusion.

11.1.3 Design Space for Menu Systems

The dimensions for the design space are location of display and menu layout. We will discuss these dimension with their possible peculiarities in this section.

Location of Display

The menu could be displayed at the following different locations.

On the work-desk If the user works with an augmented work-desk, it would be convenient to display the menu on this surface at a fixed location. Although useful, this setup is inherently stationary that is non-mobile.

On the environment In mobile scenarios it is imaginable to render the menu augmented on the environment e.g. on a tree while performing a ranger task. Although interesting, this setup requires a working 3D viewer which we decided not to wait for.

On the input device It would be possible to render the menu on tracked input device like e.g. the Chameleon Touch-Pad, but this setup has the same issues like the one mentioned earlier plus it requires the user to look on his input device while performing the menu selection which does not seem desirable from a usability standpoint at first glance. Still it seems very promising to render button and slider layout on a re-configurable device like the Chameleon Touch-Padfor very high flexibility. This setup should be evaluated at a later time when technology permits.

Head-fixed in the HMD The final option which was eventually chosen, displays the menu head-fixed in an HMD. At an former project SHEEP [41] in the chair, we learned that this is possibly a very bad choice when direct manipulation of the shown items is desired. In this specific case this is not an issue however, because we aimed at a mobile solution which would not have offered direct manipulation due to lacking precise outdoor tracking anyway.

Menu Layout

The second dimension has the following options.

Classic menu layout In floating menu systems ([14], Figure 11.5) items are usually arranged in a horizontal fashion, comparable to standard mouse controlled menus in 2D Graphical User Interfaces. This layout, slightly adapted, is also possibly well suited for direct manipulation with e.g. tangible input devices, provided no head-fixed location of display was chosen.

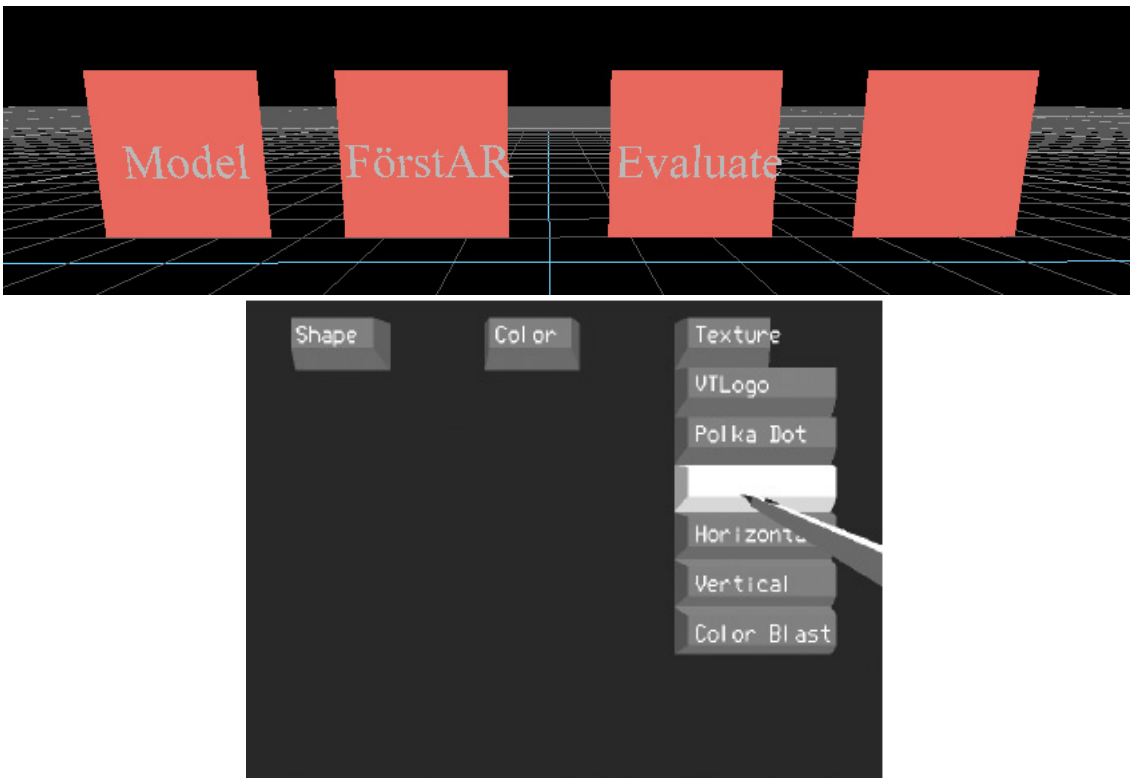


Figure 11.5: Classic menus

Pie-menu layout For a rotational input device, a circular PieMenu layout is the most straightforward mapping[61]. A menu like this might look like the mockup in Figure 11.6.

Cube menu layout A 3D menu cube layout was used in the information cube by Rekimoto. Here a nested box metaphor with semi-transparent rendering was used to visualize large hierarchical structures while still controlling the complexity of the information on the screen (Figure 11.7 [53]). A Data-Glove was used to navigate and manipulate this setup. We decided not to use such a hierarchical structure for our menus because of the big entry visibility issues, but considered a single level cube (Figure 11.7) to be e.g. controlled with a

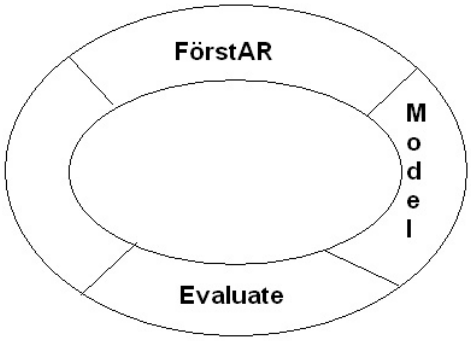


Figure 11.6: PieMenu mockup

gyroscopic, inertial tracked cube. A standard six sided cube could have a different menu entry on each side resulting in six total entries. However, unless transparency was used, not all options would be visible at once to the user. Instead if three options were sufficient, opposing sides on the cube could always share the same entry.

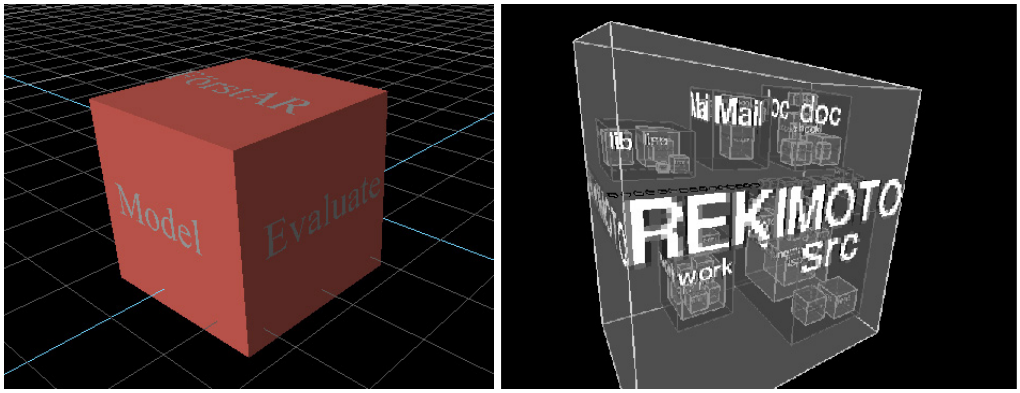


Figure 11.7: Left: Single level menu cube; Right: Nested information cube

To find the most promising combinations of layout and input devices, we created an input/widget matrix found in table 11.2.

Speech input should work equally well for both a classic and a pie menu layout. A menu showing all options, serving as something like a speech-helper seemed beneficial to usability considering a former project where we didn't have such a helper and users were confused as to which options were available [41]. Controlling a cube with speech does not seem very useful especially in cases where not all options are visible at once. Speech input would select and confirm in one single step.

We didn't see any benefit combining an inertial tracked device in combination of a button with a classic menu layout. A tracked cube like this could be used pretty well for direct manipulation of a cube menu however. A full Data-Glove would certainly be much better

Input / Widget	Speech	Gyro + discrete	Chameleon Touch-Pad slider + discrete	Custom (e.g. potentiometer + button)
Classic	++	–	++	+
Pie	++	++	+	++
Cube	o	+	-	–

Table 11.2: Input/Widget matrix for menu

suited here though, since it seems hard to highlight every side of a six sided menu cube using only one hand and a directly mapped device. By attaching an inertial tracker to the users wrist, rotational input could be leveraged for very promising PieMenu control.

Classic menus seem straightforward, and easy to use when combined with the Toch-Pad device for both slider and button input. The slider would move the high lightening and the button confirm the selection. For best usage both the slider as well as the button should be in the same area of the Chameleon Touch-Pad area. Although a PieMenu could be controlled with this setup too, it doesn't make much sense considering the not matching input metaphor, so this would probably just confuse the user. If the Chameleon Touch-Pad had 2D sliders, it could theoretically be used to rotate a cube menu on two axis but this seemed too awkward from the start to further pursue. The above can be explained with the ergonomics term of compatibility. To achieve compatibility both the hand and the widget should move in the same direction at all times.

Finally a custom rotational input device would probably be well suited for a PieMenu setup. A classic menu could be controlled with this device too, but since the metaphors don't match, it would probably just confuse the user again. Controlling a cube menu with a one dimensional rotational input device does not seem possible in a usable manner at all.

11.1.4 Decision

We thought it would be worthwhile to acquire the Intersense InertiaCube^2 in combination with a Brainbox BL-512 RS232 Bluetooth converter[5] (aprox. $300) to have a laptop independent, precise, and mobile motion tracking solution with the least amount of effort but in the end we had to cut all purchase plans due to financial issues. For the same reasons we also had to cut the idea of building our own custom device.

To meet the requirements only devices which offered good indoor and outdoor support could be considered. Although speech input might be appropriate in some occasions it cannot be seriously considered for mobile, possibly noisy, outdoor menu usage. So we had to come up with something which works with either the Chameleon Touch-Pad or the InterTrax^2 or a combination thereof.

Taking all of the above into account we decided the two most promising menu solutions to evaluate would be a classic ListMenu combined with our Chameleon Touch-Pad device and a PieMenu combined with both our Intersense InterTrax^2 for rotational input and the Chameleon Touch-Pad for discrete input.

[5] http://www.brainboxes.com/

In the PieMenu case we planned to attach the Intersense InterTrax^2 on the upper portion of the Chameleon Touch-Pad for hand rotational input.

11.2 Implementation

Here the implementation of both hardware and software aspects of the decision formed in the last section is described in detail. We will first cover the hardware considerations.

11.2.1 Hardware

As described in 11.1.4, the two hardware devices we decided to use for our menu system evaluation were the Chameleon Touch-Pad and the Intersense InterTrax^2.

Chameleon Touch-Pad

We did not actually receive the full Chameleon Touch-Pad from the Columbia university, but merely a do-it-yourself kit of the core Chameleon Touch-Pad component, which was supposed to be mounted on the palm of a half glove, as seen in the picture provided to us (Figure 11.2 on page 111).

After multiple hardware issues with the Chameleon Touch-Pad related to the cable and isolation, we decided to encase the Chameleon Touch-Pad in a small steel box and went through the effort of installing a much more solid serial port cable (Figure 11.8).

Figure 11.8: Chameleon Touch-Pad hardware improvements

Retrospectively, the history of the Chameleon Touch-Pad can be summed up as in table 11.3.

Time	State
September 2002	Received Chameleon Touch-Pad as DIY-kit
October 2002	Assembled one Chameleon Touch-Pad using the DIY-kit
January 2003	Technical difficulties have been repaired
April 2003	Re-occurring technical difficulties
May 2003	Fixed hardware instabilities by added case

Table 11.3: Chameleon Touch-Pad history

Following this we got two half gloves (for both left- and right-handed users) and attached velcro interfaces on the palm and top area of the half gloves to be able to attach both the Chameleon Touch-Pad in the palm, and the Intersense InterTrax^2 on the upper portion. For this we glued glove opposing velcro stripes on both the Chameleon Touch-Pad casing and the Intersense InterTrax^2. At the wrist area of both gloves an additional small velcro stripe was glued on to tidy up the cables of both installed devices (Figure A.7 at page 164).

Intersense InterTrax^2

Fortunately this device already was in a stable en-casing and had no hardware issues, so we could keep it as it is.

Next we cover the software aspects of the study.

11.2.2 Software

The following figure 11.9 gives a good overview of all involved software components.

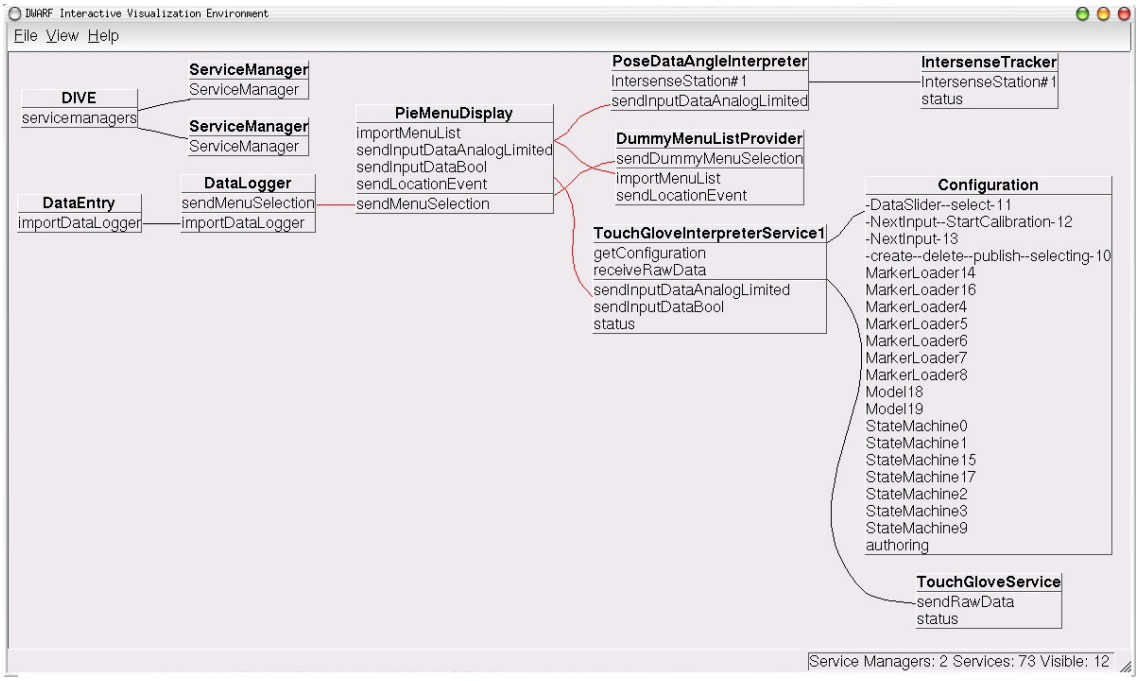


Figure 11.9: DIVE visualization of PieMenu Integration

DIVE DIVE [47] is a service used for visualizing service communication.

ServiceManager These services handle middle-ware, communication aspects of the DWARF system [39].

PieMenuDisplay This is the PieMenu display service described in more detail later in this section. It is sending "MenuSelection" events, while receiving "MenuLists" and "InputDataAnalogLimited" and "InputDataBool" events.

DummyMenuListProvider Through this service menu entries get propagated to the menu display for usability study evaluation purposes. It is also receiving "MenuSelections" from the menu display service to know which new menu to send next.

DataLogger DataLogger from the the evaluation framework is actually logging data. Here it is accepting "MenuSelection" events. See 10.1 for more details.

DataEntry This service from the evaluation framework allows the manual entry of log events. See 10.1 for more details.

TouchGloveService The driver service outputting raw data of the Chameleon Touch-Pad [76].

TouchGloveInterpreterService1 For button and slider behavior, template services, which mapped the raw data to concrete events were required [76]. Here it is mapping raw Chameleon Touch-Pad events to "InputDataBool".

Configuration We anticipated different Chameleon Touch-Pad profile needs for both menu types, so a system was installed, which loaded the desired profile into the interpreting service (*TouchGloveInterpreterService1*) on demand [62] by prior configuration (*Configuration*, [66]).

IntersenseTracker This driver service was available, detecting devices such as the Intersense InterTrax^2 on the serial port and emitting a continuous *PoseData* stream, representing the current orientation of the device [28].

PoseDataAngleInterpreter Unfortunately for the use in our PieMenu, PoseData could not be directly used. I had to write this mediator service between the *IntersenseTracker* and the *PieMenu*. It is converting the PoseData stream into an "InputDataAnalogLimited" one suitable for the menu display service.

After this short overview, details for the services specifically required for this study are described in more detail. We will first discuss *PoseDataAngleInterpreter*, followed by the menu displays (*PieMenuDisplay*, *ListMenuDisplayStatic*) and the menu entry provider (*DummyMenuListProvider*).

PoseDataAngleInterpreter

Summary This service receives *PoseData*, extracts corresponding Euler angles and maps these to *InputDataAnalogLimited* values to mimic slider like behavior with e.g. an inertial tracked device.

Needs & Abilities

Ability InputDataAnalogLimited (SvcProtPushSupplier)

Need PoseData (SvcProtPushConsumer)

It was mainly meant to interact with the previously mentioned IntersenseTracker but it can also interact with other *PoseData* emitting services such as the *ManualTracker* for debugging.

InputDataAnalogLimited events are e.g. needed by the menu displays, thus this service has to be started to control e.g. the PieMenu with a gyroscopic device.

Internal Behavior The user may specify the desired angle to map and which angle range is to be mapped between the slider range 0 and 1 by specifying command line parameters.

-Dangle Either x, y or z. By default "z" is used.

-Drange Any value higher than 0, usually below 360. By default "112" is used.

Angle "z" was chosen as default, since this is the angle the user changes when rotating his wrist with the Intersense InterTrax[2] attached to the top of his hand. By conducting pilot tests, it became obvious that a rotation range of about 90 degrees is still comfortable to the user while maintaining a range large enough for high precision. When using the default "112" range this corresponds to a rotation of about 90 degrees to go from the left most entry to the right most entry in a PieMenu with five entries.

At startup time, angle zero is mapped to the minimal slider value (0), and the angle at the border of the range to the maximal slider value (1), with other values in between mapped accordingly. The angle range is specifying how far the user has to rotate the device to go from the minimal slider value to its maximal one.

This range will never change during run-time, but auto calibration enables the user to change the borders of the minimal and maximal angle on the fly, finding the section in the full circle which she is most comfortable to rotate in. We expected most users to pick the section between 45 and 135 degrees to rotate in. The auto calibration feature even allows ranges much higher than 360 if desired.

Menu Displays

Summary We thought about implementing the menu displays in Open Inventor[6], but in the end opted to use QT from Trolltech [18], since it provided libraries to ease rapid prototyping-typing, other DWARF components already relied on QT successfully, and because the then under development being 3D viewer [30] also rendered in a QT window, meaning future integration might still be possible.

Both the *ListMenu* and the *PieMenu* share a common class for data storage, so basically only their view differs.

Needs & Abilities

Ability MenuSelection (SvcProtPushSupplier)

Need InputDataAnalogLimited (SvcProtPushConsumer); InputDataBool (SvcProtPushConsumer); MenuList (SvcProtObjrefExporter)

The *InputDataAnalogLimited* events are expected as slider values between 0 and 1 for highlighting the corresponding menu entry. For the ListMenu value 0 is the top entry, whereas for the PieMenu it's the left most one. *InputDataBool* events represent button clicks and prompt the menu service to send an *MenuSelection* event. "MenuSelection" is a generic type and can either mean a highlight on this element occurred or an actual selection confirmation. These types are kept apart by type encoding in the event body. Finally through the *MenuList* ability new menu lists can be pushed to the menu through other services on demand.

"MenuSelection" events emitted by this service are of main interest for the DataLogger in this sample usability study.

[6] http://www.sgi.com/software/inventor/

PieMenu

Summary The PieMenu display renders a number of menu entry options on a top half-circle in separate wedges for rotational input. A with sample entries populated PieMenu can be seen in figure A.8 on page 165.

Needs & Abilities This has already been covered in 11.2.2. The PieMenu only accepts the above mentioned events for both highlighting and selection of menu entries.

Internal Behavior Depending on the number of entries to display, the half circle is split up into a matching number of wedges. They are then distributed from left to right, that is the left-most wedge is the top entry and the right-most wedge represents the final entry.

Contrary to the ListMenu, the PieMenu had to be written from scratch however leveraging everything QT had to offer. The exact same font and size was used to render menu entry names like in the ListMenu.

We opted to use an upper half circle for the PieMenu, since we only intended to ask the user for a rotation of maximal 90 degrees, so a full circle would not have been a good matching metaphor. A half circle can be clearly arranged with about five menu entries, which was our aim.

The text of each menu entry was rotated and placed in the middle of the corresponding pie wedge in an attempt to retain maximal readability. It is clear however that this setup is inferior in readability especially in the middle section compared to the ListMenu, due to its nature.

Highlighting an entry in the PieMenu had to have the same effect like in the ListMenu, that is the corresponding wedge should be colored in blue plus the text color should be inverted from black to white. In the current design, rotating the hand will rotate the highlight wedge while all entries always retain their original position.

To increase readability of menu entries a different highlighting approach could be taken, that is not rotating the highlight wedge, but rotating all other entries by one position while always retaining the highlight wedge at a position with very good readability.

For example the left most wedge could always be the highlight wedge and all entries are to be rotated into this partition first before selection. Here issues arise however as to how handle elements dropping from one side gracefully.

An half-circle does not seem like a good choice using this metaphor, a full circle seems to be a better approach. For a consistent design we would need a rotational dial for full rotations however, which we did not have.

Besides I seriously doubt that this approach would yield superior usability for three reasons. Firstly the user will not only have to scan for the desired entry once, but try not to lose track of it while trying to match it with the highlight wedge when rotating the dial. Secondly this approach will not really help readability of single menu entries, unless the user wants to rotate all entries in the highlight wedge for maximum readability before even scanning for it in the overview. Finally it seems more intuitive to bring the highlight to the desired entry than to bring the desired entry to the highlight at first glance.

Because of the above mentioned reasons, we decided to stick with the original setup but still it might be worthwhile to remember the other design approach, once we have more rotational input devices to experiment with.

ListMenu

Summary In the ListMenu items are arranged in a standard vertical list. The ListMenu with sample entries can be seen in figure A.6 on page 164.

The DIVE [47] visualization in figure 11.10 shows in an overview how all services are integrated for a ListMenu usability study.

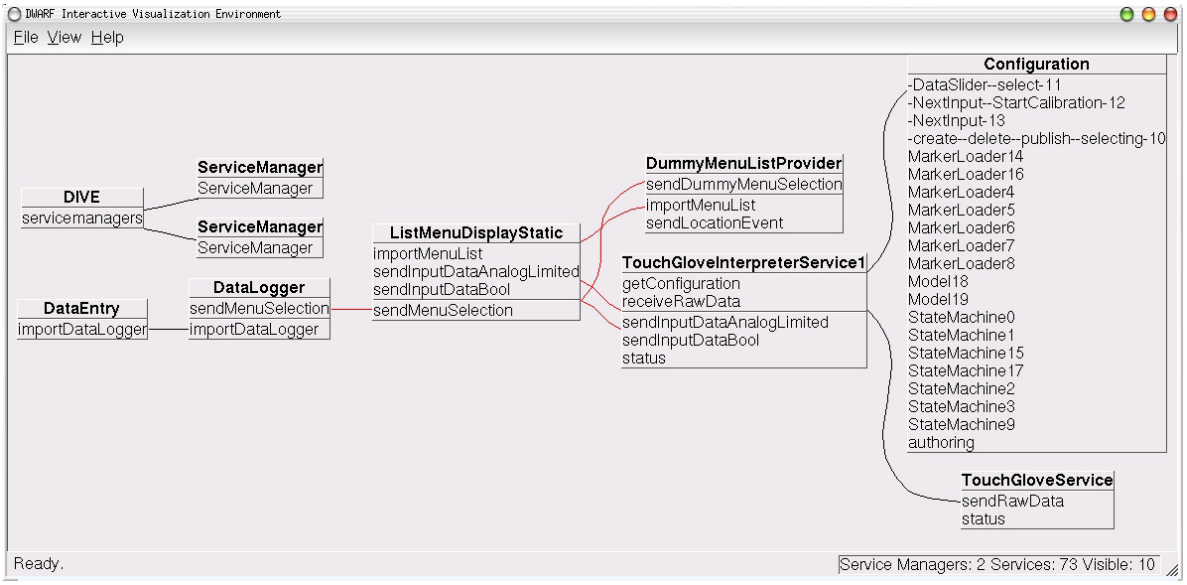


Figure 11.10: DIVE visualization of ListMenu Integration

It should not be surprising that compared to the PieMenu overview picture both the *IntersenseTracker* and the *PoseDataAngleInterpreter* are missing. All other services are the same. This time however a different profile was loaded into *TouchGloveInterpreterService1* by means of *Configuration* so that it now provides both the "InputDataAnalogLimited" and the "InputDataBool" need to the menu display service *ListMenuDisplayStatic*

Needs & Abilities This has already been briefly covered in 11.2.2. Entries in the ListMenu can be either highlighted and selected by mouse clicks or via *InputDataAnalogLimited* events for highlighting and *InputDataBool* for selection.

Internal Behavior Implementation of the ListMenu itself was fairly straight forward since QT already offered a list box library construct we could leverage. It supported highlighting and selection already on its own basically.

The button at the bottom is used for mouse operation, which the *Selector* service in ARCHIE relies on [62]. The needs for the above mentioned Chameleon Touch-Pad events are registered and a bigger font is used when the command line parameter

```
-Dusability=yes
```

is passed to the ListMenu executable.

Menu Entry Provider

Summary For the study we had to populate the menus with entries for the participants to select. To accomplish this an extra service, the *DummyMenuListProvider* was written. Menus were used where one single entry was open for selection, while all other entries were disabled. The user would then have to try to select the open one.

Needs & Abilities

Ability MenuList (SvcProtObjrefImporter)

Need MenuSelection (SvcProtPushConsumer)

This service must have the ability "MenuList" to be able to push new menus to the menu displays. Finally it required a need for "MenuSelection" events from the menu display to know when to advance to the next step, that is when the user selected the "open" entry of the former step.

Internal Behavior The list provider can be set to offer different sets of menu entries for selection through which it will cycle continuously via command line:

-Dlist An int value between 3 and 7. By default "5" is used.

This number specifies the number of entries to display in one menu. Per default five menu entries will be pushed to the menu display. Selecting about 30 times in either menu takes about two minutes, which seemed like an appropriate time for one single task.

30 numbers between zero and one were randomly picked, where each number depicts which entry should be the one open for selection in this task step. The open entries were always named "$SelectMe! < x/30 >$", were x represented the current step. The final open entry was "$SelectMe! < Final >$". The disabled entries were named "$Disabled$" at all times.

It was important to maintain this random order through all task repetitions since we wanted to collate user speed ups in menu usage solely to learn effects of the interface and not to a possibly "better" open entry sequence in a later run.

Please refer to the appendix to see the exact number sequence.

11.3 Study

This chapter describes the conduction of the study and then gives a detailed report about the results.

11.3.1 Conduction

Nine participants were acquired for this study. All but three were internal, pretty much expert users who mostly knew what the evaluation was about prior to the study. No special efforts were undertaken to win these participants. To save time I mostly did "hallway testing", that is I tried to evaluate anyone who went by the hallway and had the time to participate after everything was prepared and set up.

The orientation script found in Appendix A.3.1 was read to each participant.

For detailed review, each participant was given a printout of the general usage guidelines (Appendix A.3.2) plus the combined background and pretest questionnaire (Appendix A.4.1) for reading and completion prior to the study.

The chosen tasks were very simple, requiring only menu selections. Please refer to the above mentioned materials to see what exactly was asked from the participants. The two main tasks "ListMenu" and "PieMenu" were each to be repeated two times for a total of three runs to observe learning effects. 30 selections in each task run were required from the participant, since prior rough measurements showed that one such run takes about two minutes on average, which seemed adequate for a single task. All in all the evaluation process took about 30 minutes, including the completion of all questionnaires. To avoid cross-learning effects between the main tasks "ListMenu" and "PieMenu" the presentation of these tasks was always alternated in-between participant evaluations.

After each menu type task completion, one copy of the posttest questionnaire (Appendix A.4.3, already summarized) was given to the participant. So the main posttest questionnaire was filled out twice, once for each menu type. The second posttest questionnaire was combined with the overall questions (Appendix A.4.4, already summarized).

All questions on the posttest and overall questionnaires had free text answer fields in the form of
Explain: _____

Before releasing the participant, questions specific to the participants performance which were of particular interest were posed verbally.
To summarize the following evaluation checkbox-list was used:

□ Greet and read the orientation script to the participant

□ Pass out general usage guidelines and wait for the participant to finish reading

□ Pass out combined background and pretest questionnaire

- □ Start first menu type task series
- □ Pass out posttest questionnaire
- □ Start second menu type task series
- □ Pass out posttest and overall questionnaire
- □ Pose verbal follow up questions and debrief participant

The evaluation was conducted in a lab which is depicted in figure 11.11.

Figure 11.11: Sample Study Setup

The white cross in the front depicts the place where I was sitting as the evaluation monitor. The white cross on the floor in the back marks the spot where the participant was standing. The distance between these two points was chosen at about four feet to not be too close to the participant while she performed her tasks. The participant was asked to look in the top left corner in the room so I could clearly see both facial expressions as well as handling of the input devices. Observers were tolerated if they remained far enough in the background and didn't interfere in the study in any way.

The HMD of the user was attached to the laptop on the round table which showed the menu the user was seeing on its own display for me to observe, solving the action visualization problem in a very trivial manner. The HMD was set to maximum transparency mode. It doesn't make much sense to use a wearable device sitting or with the display on a workdesk monitor, so the HMD was mandatory additional to standing upright.

Both the Chameleon Touch-Pad and the Intersense InterTrax^2 were connected to serial ports on the computer to the left which was also used by me to enter manual logging data via *DataEntry*.

I prepared two gloves, one for each hand so users could pick the one which they preferred depending if they were right or left handed.

A user conducting the "PieMenu" task is shown in figure 11.12.

Figure 11.12: Participant During PieMenu Task

The setup for the "ListMenu" task changed from the pilot to the final study so it is not described in further detail until the next section.

11.3.2 Results

Here the pilot study with one participant is covered because it had interesting, serious implications on the final study setup. A prototype history of the "ListMenu" task setup, resulting in several improvements, is given together with the rationale for changes.

Pilot Study In the pilot study the "ListMenu" task was also executed with the Chameleon Touch-Pad installed on the palm of the users hand just like already seen in figure 11.2 on page 111. Because of a similar, already existing setup [13] we were very focused on this assembly from the very start.

With the Chameleon Touch-Pad attached to the palm, the participant had to bend over one of her fingers to slide on it for highlighting. Finally to select an entry, the finger was to be lifted shortly and put down again for the button press. Since we only required one slider the whole area of the Chameleon Touch-Pad was configured to be a slider in effect allowing the participant to use any finger he desired.

The original idea of this setup was to allow four sliders at once, one for each finger. There is not much margin for movement in the horizontal area when bending your fingers to your palm like this which might make you think this is a suitable setup. I had my doubts with this

setup from the very beginning though. It might seem "smart" at first but its usability never struck out to me as any good.

This type of finger movement is awkward in any case and prone to causing cramps. By experimenting with it on my own, it became clear it is pretty hard to move the finger all the way down to highlight the bottom most entry while only touching the surface with the tip of the finger. The further the finger slid down, the more likely it was to touch the upper surface with the flesh of the remaining finger in effect confusing the Chameleon Touch-Pad interpreter service considerably. The new contact was interpreted as a new slider event, messing everything up. The problem was certainly reinforced by the added case to the Chameleon Touch-Pad, which considerably increased its overall thickness. Additionally the velcro contacts between the glove palm area and the Chameleon Touch-Pad increased the height on the palm even further.

For the pilot study I went through great lengths to improve on this while still maintaining the initial setup. It was obvious that usability could be increased with two distinct methods.

Apply Pressure If the other hand was used to press the Chameleon Touch-Pad down to be much closer to the palm, it was much easier to slide on it with a finger of the glove hand.

Add Wedge By adding an about 45 degree wedge between the Chameleon Touch-Pad and the palm, finger sliding also seemed to be much easier. The finger didn't have to go all the way down to the palm surface anymore, but could remain at a much more comforting angle.

I saw no way of using the "apply pressure" method so I went with the wedge. To realize this, yet another velcro area was glued to the Chameleon Touch-Pad which was used to connect a short wooden block to act as the wedge. Due to their nature, all velcro interfaces were not very secure or hard fixed. I'm sure much better usability could have been achieved if the Chameleon Touch-Pad was woven into the glove.

So although this wedge did improve the usability of the "ListMenu" setup somewhat, I was still not really happy with it but started to conduct the pilot study anyway to get some first feedback. It was very revealing to say the least.

The pilot user conducted the "PieMenu" task series first, after which he performed the old "ListMenu" setup.

Let us take a look first at the individual task times in figure 11.13.

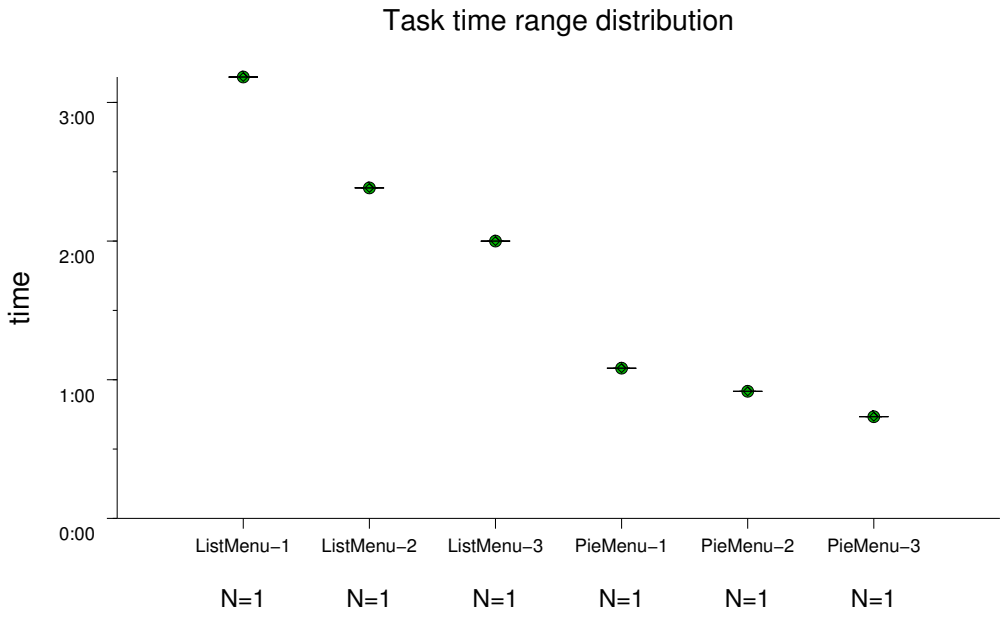


Figure 11.13: Pilot Study Visualization, Task Time Range

This visualization clearly shows learn-effects, both in the "PieMenu" as well as in the "ListMenu" tasks. But combined, the task time difference looks very drastic. The line which can be plotted through these points is almost asymptotic with the "ListMenu" tasks at the top and the "PieMenu" ones at the bottom.

The participant complained verbally quite a lot during the "ListMenu" task, which can be visualized pretty nicely in the second task run in figure 11.14.

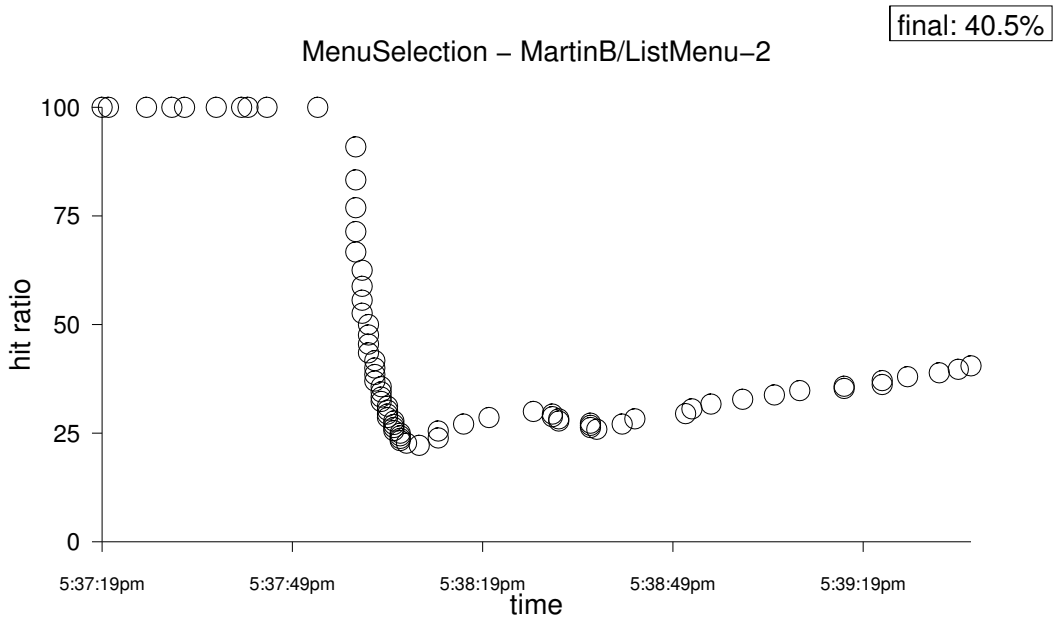


Figure 11.14: Pilot Study Visualization, Hit Ratio

Although the hit ratio was very good at first, it dropped very low with about 40 repeated misses (total 44 misses). The user had serious trouble trying to select one single entry for the reasons mentioned above.

The hit ratios for all tasks are shown in the following table 11.4.

Task	Hit Ratio	Time
ListMenu-1	78.9%	03:11
ListMenu-2	40.5%	02:23
ListMenu-3	96.8%	02:00
PieMenu-1	73.2%	01:05
PieMenu-2	75.0%	00:55
PieMenu-3	85.7%	00:44

Table 11.4: Hit Ratio for Pilot Study (Single User)

It's interesting to note that task "ListMenu-3" had such a high hit ratio with only one miss, but at the cost of time. I already suspected the reason behind this here but it didn't become totally obvious until the final study where it is described.

The pilot study also made one bug in the `ListMenu` implementation obvious which did

falsify the result a bit. Whenever the user performed a successful selection, a new menu list was send to the menu where the now open entry was placed somewhere else. However each time the `ListMenu` got a new menu entry list, it automatically highlighted the top most entry independent on previous selection.

This is inconsistent with expected touchpad behavior to the experienced notebook touchpad user. This insight was extracted from the posttest questionnaire concerning the "ListMenu" task.

Although the study already started with one participant, but due to above very serious findings, I decided to move these results down to pilot study level and not to continue with the study until these major flaws were fixed. It would not have made much sense to continue the study like this. After all a study only makes sense if you are not certain about the outcome in advance.

First I fixed the highlighting behavior of the `ListMenu`, meaning it now remembered which entry was last highlighted and will keep the highlight at this position even after a selection succeeded and a new menu was rendered.

At this point I also rethought the whole setup of both the "ListMenu" and "PieMenu" tasks.

PieMenu Setup Here the glove setup made a lot of sense. Only with the hand such precise rotations for menu highlights are possible. The Chameleon Touch-Pad in the palm area of the rotating hand also seemed very reasonable, since it was only used for button clicks, with little room for improvement.

The results of the "PieMenu" task in the pilot study were also consistent throughout like expected. So I saw absolutely no reason to modify this setup. I still considered this though since I had to reset the study anyway.

ListMenu Setup I rethought why did we used the glove here at all [13].

We only needed one big slider and one big button. No multiple sliders and definitely not four. So there is actually no good reason to force four channels on the Chameleon Touch-Pad for sliding by attaching it to the palm of a glove.

Like mentioned above, I was always sceptical about this setup. Wearing a glove is a big disadvantage in any case. There is no big advantage here using one like in the "PieMenu" task, which would be perspicuous to the participant.

There is no reason not to consider attaching the Chameleon Touch-Pad at a completely different location like on the hip or on the breast. This would still only require one hand for operation while actually keeping the hand totally free when the menu is not in use providing a big advantage to this setup. It would make sense to modify the study by giving the participants further tasks while using the menu which require both hands. There is a slight chance the "ListMenu" might even overcome the "PieMenu" in these cases. This requires further research.

I had to decide on one fix anchoring point however. Of course it would have been possible to just allow the user to attach the Chameleon Touch-Pad everywhere she wanted, but this

would have resulted in a very uncontrolled evaluation environment. I do think a future study should be made on this however where different anchoring positions are evaluated to find the most usable one. This study would focus on the "ListMenu" alone but with different attachment points like forearm, hip, breast, thigh etc.

We did have a convenient backpack which had a velcro attachment point at the breast area out of the box (See figure 11.15).

Figure 11.15: Backpack for Chameleon Touch-Pad Fixation at the Breast Area

A quick evaluation of this setup showed major improvements in the Chameleon Touch-Pad usability for both selection and highlighting in the "ListMenu" task. The final study was conducted with this setup as shown in figure 11.16.

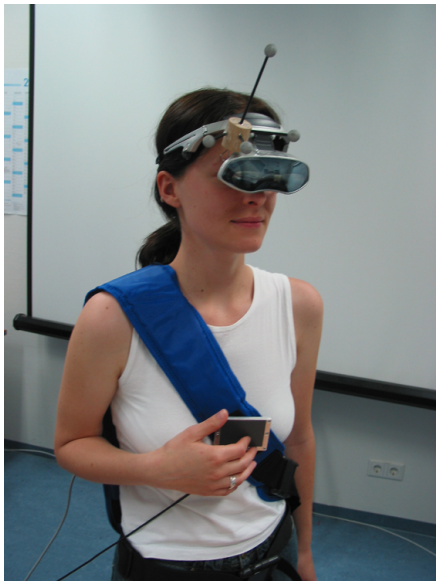


Figure 11.16: Participant during ListMenu Task

To fully exploit the changed setup, the Chameleon Touch-Pad profile was further modified to rotate the slider orientation 90 degrees so the full length of the Chameleon Touch-Pad surface could be leveraged for menu highlighting, allowing more precise movement.

The participant was allowed to place the Chameleon Touch-Pad on the velcro surface at their orientation of choice, although it was suggested to attach it like shown in figure 11.16 to minimize hand-strain, even though the metaphor matches better when it is attached upside down.

Final Study

After settling on the final setup, the study was conducted with all remaining participants. The results are presented now. Please see Appendix A.4 for a summary of all returned questionnaires with my added comments for statistical considerations and all the answers.

To quickly summarize, all participants where in the 18-29 age group, with two females and seven males, high degree of education (usually computer science) with an average of 9.83 years of computer experience. The participants used computers on a regular work day for 6 hours. The exposure to Augmented Reality and related technologies was on average very low. Like mentioned earlier, there were a number of internal testers with expert knowledge (4). These internal testers were actually members of the ARCHIE team, so most of them even got to train both devices prior to the study. Two more participants were expert computer users while the remaining three were real least knowledgeable users added to the sample. Acquiring these three took most effort for the study since they usually do not walk by the hallway next to a computer science laboratory.

Three novices are not really sufficient to split the participants into two groups dependent on their prior experience. This is why all following visualizations take all users into account. Still this fact should be remembered for the following argumentation.

The analysis is split into three parts. We will first only cover observations which can be made by looking at the visualizations of logging data. Anyone with access to general evaluation setup information, the log data and above statistical knowledge could deduct these without looking at the posttest questionnaires or without having attended the actual study.

After doing so I will add more semantics to these first results by adding what I observed by monitoring during the study.

Finally the remaining results are provided by analyzing the posttest questionnaires in detail.

Closing this chapter an executive summary quickly provides the main results in an overview.

1. VISUALIZATION OBSERVATIONS

See figure 11.17 for a quick overview over the task time range distribution.

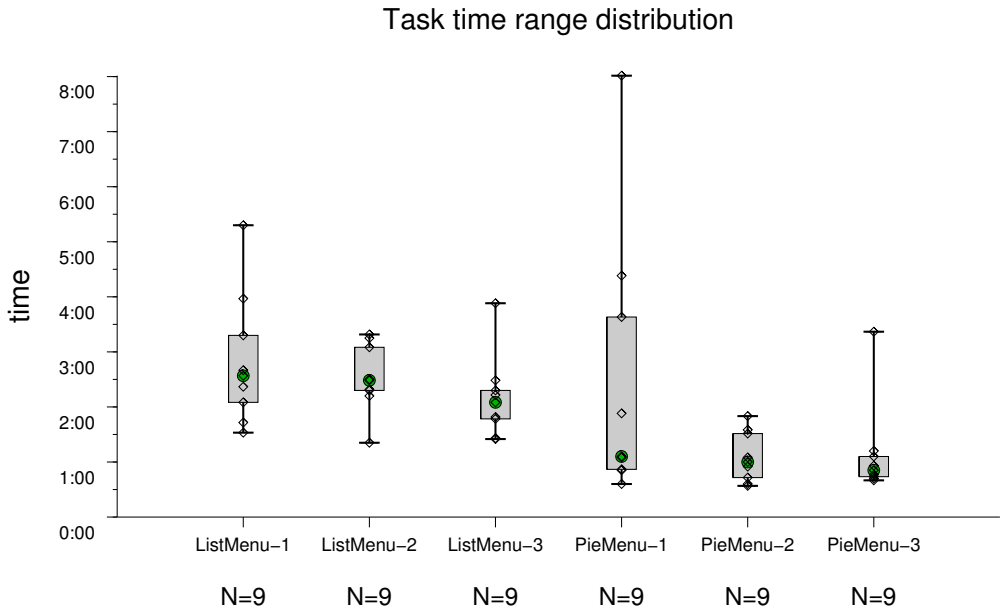


Figure 11.17: Task Time Range - Median (The biggest dot indicates the median time for each task while the box-plot extends to the 25th and 75th percentile. The error-tails extend to the border values. The smaller light dots show the individual task completion times of all participants.)

The border values have been rather extreme for e.g. “PieMenu-1”, which is why a mean and standard deviation visualization is probably more helpful (Figure 11.18).

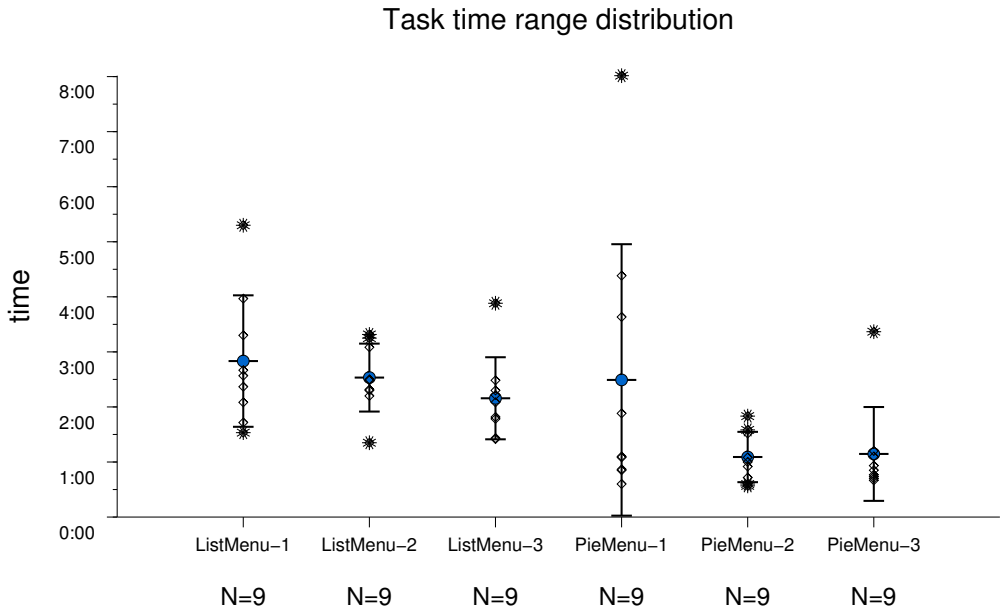


Figure 11.18: Task Time Range - Mean and Standard Deviation (The biggest dot indicates the mean times and the error bars extend to the standard deviation. The smaller light dots show the individual task completion times of all participants. The stars denote task completion times outside of the standard deviation.)

It is obvious that the PieMenu benefited tremendously from learning effects while the ListMenu had some too.

Both visualizations are summed up in table 11.5.

Task	Minimal	Maximal	Mean	Standard Deviation	Median
ListMenu-1	1:32	5:18	2:50	1.19	2:34
ListMenu-2	1:21	3:19	2:32	0.61	2:29
ListMenu-3	1:25	3:53	2:94	0.74	2:05
PieMenu-1	0:36	8:01	2:29	2.46	1:06
PieMenu-2	0:34	1:50	1:05	0.45	1:00
PieMenu-3	0:40	3:22	1:08	0.85	0:51

Table 11.5: Sample Study Task Time Range

The tasks were very simple, so it could not be avoided that some internal testers clearly tried to compete to set a time record.

These are not ideal evaluation conditions but this knowledge can be exploited by looking at the overall minimum task times.

The best PieMenu run (0:34) was by factor 2.38 faster than the best ListMenu run (1:21), giving it a clear edge under expert conditions. This quickest ListMenu run was still slower than both the mean and median times of the second and third PieMenu task runs.

Looking at the overall visualizations however both menu types are surprisingly close.

Lets take a look at the numbers of the average absolute misses at selections for the ListMenu in figure 11.19.

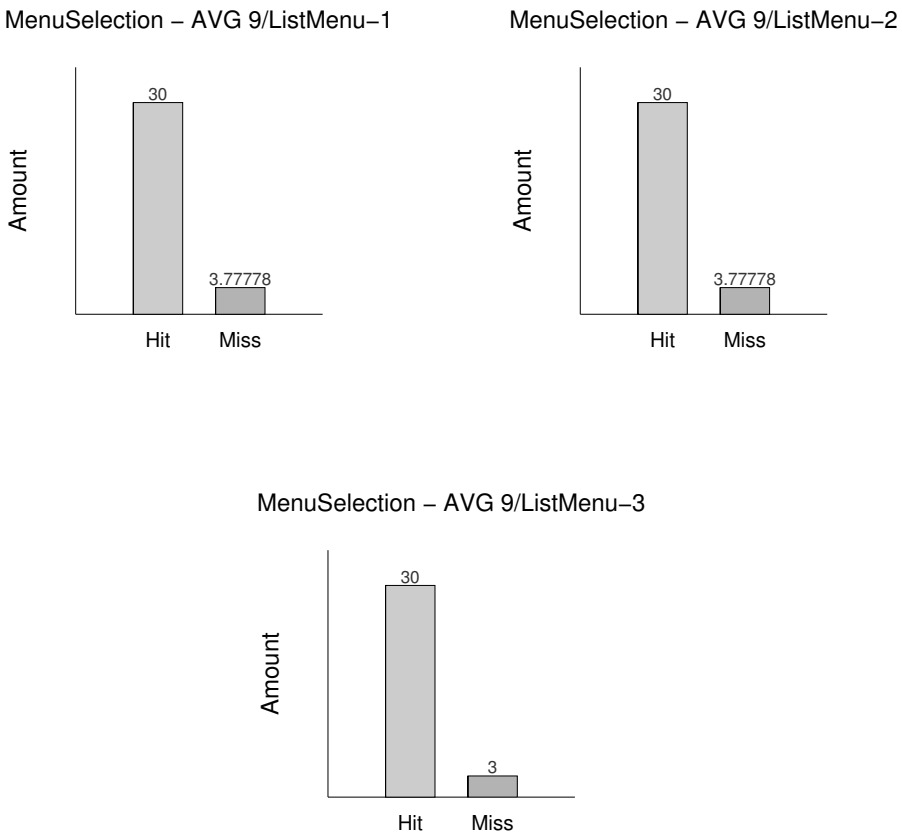


Figure 11.19: Average Absolute Bars ListMenu Task

As can be seen, the error number was pretty much constant through all three task runs. The same statistic is shown for the PieMenu in figure 11.20.

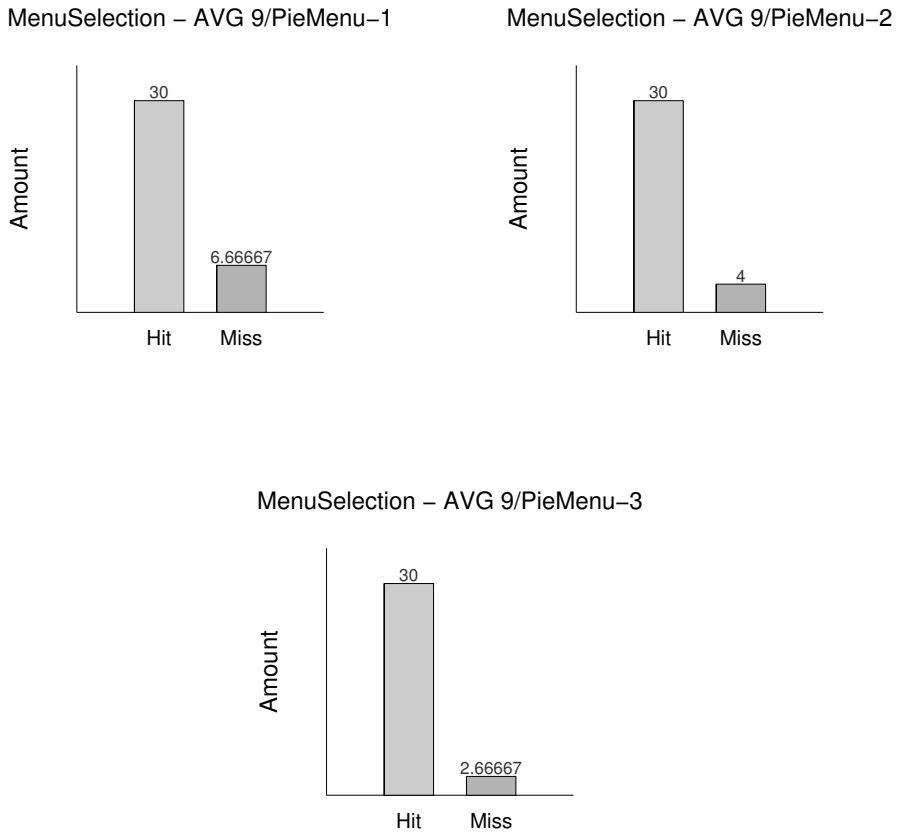


Figure 11.20: Average Absolute Bars PieMenu Task

The average number of errors is here actually a little higher at first with a steady decrease resulting in an all time low of only 2.6 errors of 30 selections.

Overall the difference between both menus is not noteworthy though. So why did the ListMenu tasks take so much longer? Hit ratios where computed for each task and put in an overview in figure 11.21.

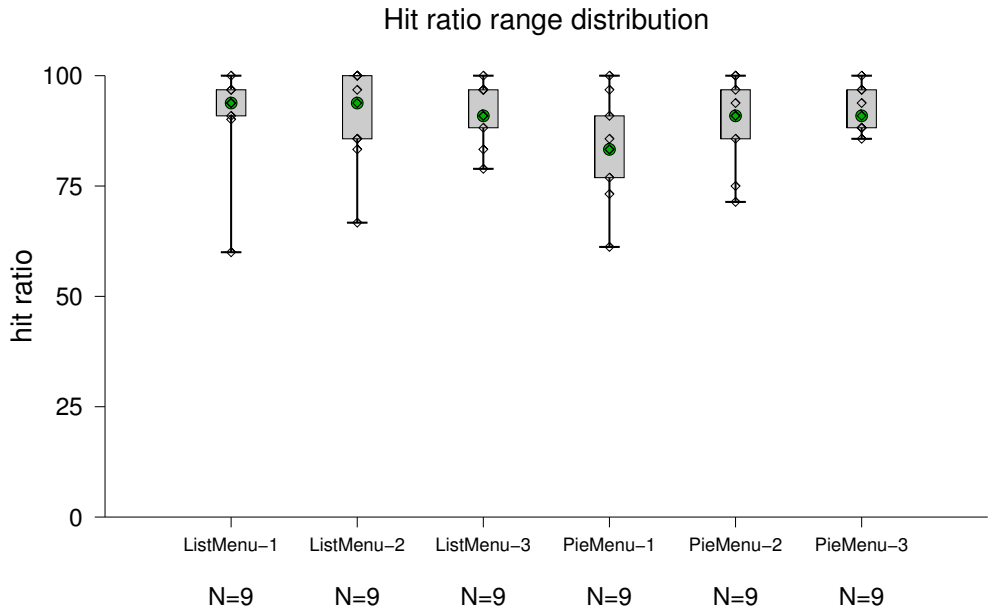


Figure 11.21: Hit Ratio Range Distribution - Median (The biggest dot indicates the median time for each task while the box-plot extends to the 25th and 75th percentile. The error-tails extend to the border values. The smaller light dots show the individual task completion times of all participants.)

The difference between both menu types at average hit ratio range distribution is negligible. This still does not answer why the ListMenu tasks took more time.

If we take a look at intra-subject hit-ratios we get closer to the answer (See table 11.6)

Participant	ListMenu-1	ListMenu-2	ListMenu-3	PieMenu-1	PieMenu-2	PieMenu-3
Participant 1	93.8%	96.8%	96.8%	73.2%	75%	85.7%
Participant 2	96.8%	100%	90.9%	96.8%	90.9%	93.8%
Participant 3	60%	83.3%	96.8%	85.7%	90.9%	88.2%
Participant 4	100%	100%	78.9%	83.3%	100%	88.2%
Participant 5	93.8%	85.7%	96.8%	90.9%	93.8%	96.8%
Participant 6	90.9%	85.7%	90.9%	61.2%	85.7%	90.9%
Participant 7	90.1%	66.7%	83.3%	100%	100%	100%
Participant 8	96.8%	100%	100%	83.3%	71.4%	88.2%
Participant 9	93.8%	93.8%	88.2%	76.9%	96.8%	96.8%

Table 11.6: Sample Study Task Hit Ratios

Astonishingly participant 8 and 4 had better hit ratios in the ListMenu tasks than in the

PieMenu ones.

One trivial reason why the PieMenu tends to have a worse hit ratio is its multi-modality. Rotational movements are used for highlighting while clicking is used for selection. Participants could have easily constantly clicked, generating lots of misses, while rotating their hands eventually hitting the correct entry when the arm is the correct rotation angle.

This is not possible with the ListMenu since here both highlighting and selection takes place on the same device.

Still lets take a closer look at the second ListMenu run of participant 8 in figure 11.22.

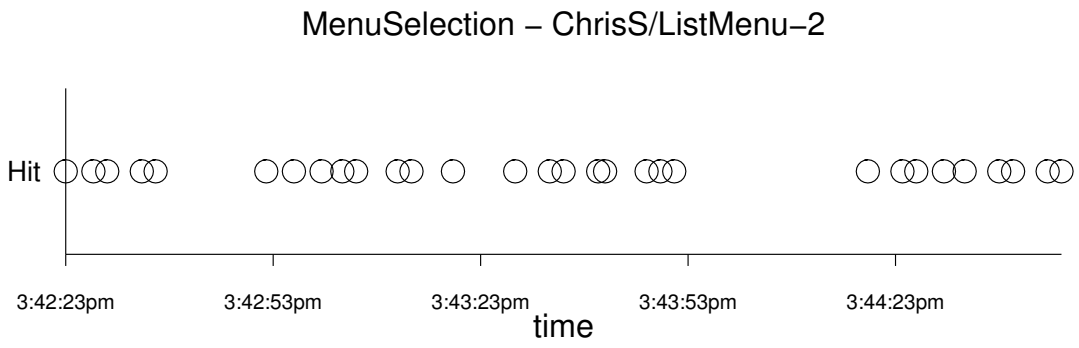


Figure 11.22: ListMenu-2 Absolute Errors Participant 8

There are big time gaps in between some of the selections. Up to participant six I only logged actual "MenuSelection" hits and misses. I then realized however that the "MenuHighlight" hits and misses should be logged too to properly visualize the problem. This new visualization was applied to the above visualization in figure 11.23.

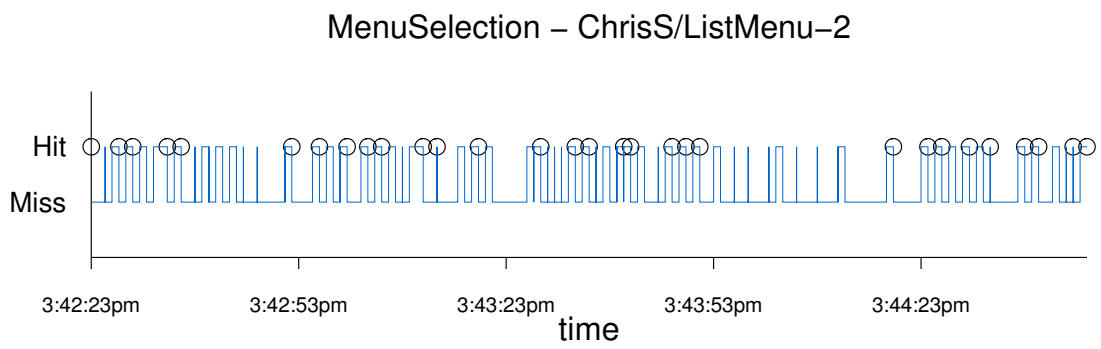


Figure 11.23: ListMenu-2 Absolute Errors with added Highlight Timeline Participant 8

The added line shows when the correct item was highlighted and the dots indicate when it was actually selected. This new line clearly shows that the correct item was highlighted often, even repeatedly, but no selection followed. Obviously there were big problems in the selection process. This result for the single participant was actually typical for all "ListMenu" task runs. So we do now have an idea why the "ListMenu" tasks took so much longer to complete. The initial hit ratios for the "ListMenu" were clearly misleading at first. So although this particular participant finished this task run with a hit ratio of full 100% which should be an indicator for precision, clearly it was much worse.

For some participants this problem even lead to "misses" as shown in figure 11.24.

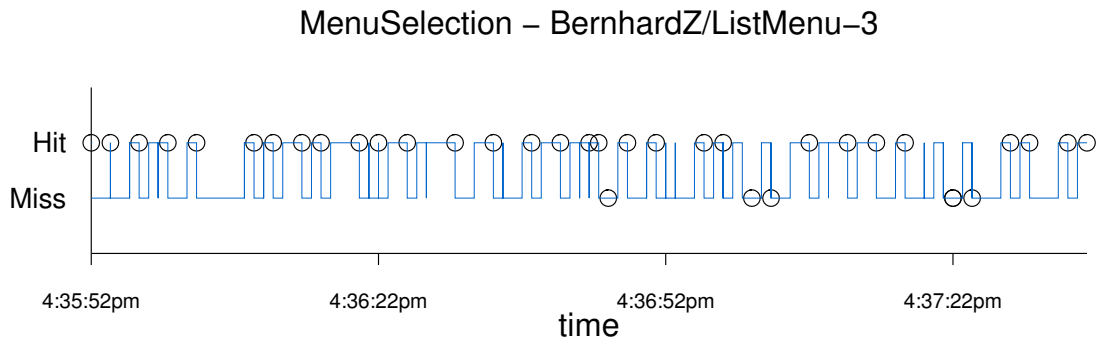


Figure 11.24: Sample Study Visualization, ListMenu-3 Absolute Errors with added Highlight Timeline Participant 7

While the previous participant often hovered at the correct item but didn't select it until after multiple more highlight attempts, this participant actually didn't only lose the highlight but also immediately afterwards executed a selection which resulted in recorded "misses".

We can conclude that the participants obviously had trouble using the Chameleon TouchPad to properly select entries and while trying so they often moved the highlight to a wrong position instead of confirming the previously correct highlight.

In the following section I will add my own observations during monitoring, which delivers much richer conclusions than what can be drawn from relying on data visualizations alone.

It is future research how to improve the data visualization and measuring methods to gain more results merely relying on this. One helpful visualization in this specific case which was not implemented due to time constraints would be one showing the total time spend with the highlight in the correct position without clicking in effect visualizing the time in error.

2. STUDY MONITORING OBSERVATIONS

I noticed a couple of trends.

ListMenu annoying, PieMenu "cool" It was a clear trend that most participants liked the PieMenu for its sheer "coolness" factor while they were mostly annoyed with the ListMenu tasks and glad when they got rid of it.

ListMenu highlight too imprecise The highlighting of ListMenu entries was very imprecise. Participants often were only slowly closing in on the entry they wanted, with

the highlight bar jumping around it. Very few participants managed to go straight to the entry without losing the correct highlight again for multiple directly following selections.

This was mostly due to lag issues it had. The time which passed after sliding on it and until it actually responded on the screen was too slow.

When lifting the finger of the Chameleon Touch-Pad often a new slider movement was registered which changed the highlighted entry before the participant had the chance to confirm it by tapping the pad once more. This was because the finger wasn't lifted straight away from the pad but "rolled off".

ListMenu highlight bar movement unexpected A very serious flaw was the implemented highlight bar movement which was mapped to absolute values on the Chameleon Touch-Pad. It was not possible to move the bar to the bottom most entry with multiple quick relative movements at the top of the Chameleon Touch-Pad. Instead a slider movement had to be recognized at the physical bottom of the Chameleon Touch-Pad which is inconsistent to typical notebook touchpad behavior.

Many users expected this and tried to perform relative movements without success. Some understood the mapping after a while and tried to speed up the highlighting by starting a slider movement already in close vicinity to the target area.

If they failed to highlight the bottom entry, most participants slowly moved their finger from the top to the bottom multiple times which was therefore probably the hardest entry to highlight, although I think it should be said all entries in the ListMenu caused problems so all were about equally hard to select.

Clearly this behavior poses a serious flaw and must be fixed.

PieMenu highlight very precise The highlighting of the PieMenu was overall very good and precise. Most users could go straight to the entry they wanted and move on to the next.

Chameleon Touch-Pad button clicks too hard Since the Chameleon Touch-Pad interpreter had to recognize both slide movements and button clicks, timings had to be set, defining how long the pad had to be touched to trigger either a button or the beginning of a slider movement. In the study the button clicks had to be executed very quickly by only tapping the pad very shortly or a slider movement was recognized. This behavior was identical for both the PieMenu as well as the ListMenu.

It was very obvious that these button clicks were way too hard to perform. Although the PieMenu had very good highlight precision, its usability was considerably lowered by the hard to do button clicks.

Some users always executed double clicks on the pad to make sure one click got through in accordance to typical GUI usage conditioning.

The confusion why clicks sometimes got through and sometimes not, was big for many participants.

In general the differentiation between slider movement and button click on the ListMenu was one of its biggest problems. Very often the user tried to perform a click but instead his click was interpreted as a slider move, so the highlight was moved to another entry not intended by the participant.

Calm, slow usage "won" It was interesting to note that participants who used the devices in a calm way often were superior to others who tried to compete for best time

while rushing.

This was especially obvious in the PieMenu tasks. Calm users rotated the device slowly in the intended, comfortable angle range, while others wrenched their arms trying to get the border items by rotating too quickly. This is also why border entries were the hardest entries to highlight for most users in the PieMenu tasks.

PieMenu arm position It was noticeable that users tended to raise their arm while often maintaining an angle of about 45 degrees in their elbow joint the longer they performed PieMenu tasks.

One participant commented on that by saying he felt that his arm was better tracked by the Intersense InterTrax^2 holding it like that.

Users who rotated out of their wrist had much less trouble with wrenching their arms than participants who rotated out of the whole arm.

A few participants would have liked to support their rotating arm with the other one.

PieMenu metaphor The usage metaphor was mostly clear. But one user did misunderstand it at first and did something very unusual. Instead of rotating his wrist, he at first slid his hand from left to right which actually also moved the highlight wedge, but mirrored.

Auto-calibration was rarely exploited by participants. When they had to highlight the right most entry while having the arm already at a very high rotation angle to the right, they wrenched their arm even further resulting in very awkward movements instead of quickly moving the highlight to the far left to recalibrate the left border.

One user commented on this and said he knew about it but thought it would be too troublesome.

ListMenu setup The Chameleon Touch-Pad location at the breast did not seem to be the best choice. Especially female participants were not too happy with this setup. It should be future research to evaluate a belt or thigh positioning instead.

Some users preferred to attach the Chameleon Touch-Pad with a 90 degree rotation, so up would be directly mapped to up and down to down, matching the metaphor more closely, although it resulted in a hand movement which is probably more straining over longer use.

PieMenu setup bad for small hands It was already mentioned earlier that the Chameleon Touch-Pad construction on the palm of the hand had a pretty large thickness due to various reasons.

The exceptionally long task completion time in one of the first PieMenu task runs, depicted in figure 11.17 on page 136, was due to this problem in combination with the hard button clicks since this user had to learn the clicks with the PieMenu.

The user first thought that she had to touch the pad somewhere in the middle to execute button clicks but had serious trouble reaching the pad area with her comparatively short index finger without using her small finger to hold the Chameleon Touch-Pad closer to her palm. As can be seen in the visualization above, this was not a problem anymore in the next run because she realized she could tap it anywhere, also on top, to execute button clicks.

Technical difficulties There have been a number of minor technical problems during the study which falsified the results to a small degree.

In some cases the HMD shut down automatically because it warned the user about the two hour running time to not over-strain his eyes requiring a confirmation. A couple of times it was necessary to re-attach the HMD since it slid down the participants head. One participant even insisted on holding her HMD with her other hand.

A couple of times there were calibration, drifting issues with the Intersense InterTrax^2 resulting in a need to recalibrate the device using auto-calibration in the PieMenu tasks.

Further results were drawn from the analysis of the questionnaires found in A.4.

3. QUESTIONNAIRES ANALYSIS

The questionnaires answered a couple of more questions but mostly added new ideas for modifications and consecutive evaluation.

PieMenu gauge meter There was a suggestion for showing a gauge meter in the PieMenu display showing exactly where the rotation currently is registered in the wedge. Without this indication the user never knows exactly how much more he has to rotate to highlight the next entry. This should be evaluated in future studies.

PieMenu auto centering Some users thought it might be appropriate to automatically move the highlight to the middle position when the Intersense InterTrax^2 has not been moved for a short while as a form of "reset". Timing this properly without confusing the user would be another topic for further studies.

PieMenu selection behavior A few users complained that they moved the highlight wedge to the next one when they tried to perform a button click that is they had trouble keeping their hand at a fix rotation while performing the button click. A suggestion was to pick the wedge, which was highlighted a milli-second before the button click, for selection, instead of the current one to counter balance this.

Another suggestion had the idea of using speech instead of button clicks for selection.

PieMenu drive control There already was a slight trend in participants wanting to support their rotation arm. One participant made the interesting suggestion to have a two handed rotational control similar to driving a car. Spontaneously I don't see a good way to realize this with wearable devices however.

Personal calibration of both menu types Some users would have preferred a smaller or a bigger PieMenu movement range, while others would have preferred a smaller physical ListMenu slider on the Chameleon Touch-Pad.

Personal configurations of slider or PieMenu rotation sensitivity should be offered just like it is common for users to fine tune the sensitivity of their computer mice. Especially users who were used to a very high mouse sensitivity were annoyed by the big ListMenu slider.

Fonts too small There have been a couple of complaints about the fonts being too small in both menus. By evaluating the questionnaires it became obvious that many users didn't actually read the menu entries but just selected the "bigger looking" entry.

The $< SelectMe! < X/Y >$ entry was always visually easily distinguishable from the smaller $< Disabled! >$ entries.

In a future study random names should be picked to properly measure entry readability levels.

Fast / Slow movement A few users reported the highlight bar sometimes moved fast and sometimes slow in the ListMenu tasks, due to lag. The slider worked best when users slid slowly on it resulting in pretty good reaction times. If the finger was slid over the area in a faster fashion the interpreter of the Chameleon Touch-Pad usually didn't even pick it up as a slide movement at all.

The same impression was had in a couple of PieMenu tasks probably due to drifting issues of the Intersense InterTrax^2.

Participant Preference Finally all nine participants preferred the PieMenu over the ListMenu.

It was a good idea to both offer categorized answers and free form answers in all questions in the posttest questionnaires. Some participants preferred to check only the quick rough category while others preferred to elaborate further in the free text forms.

4. EXECUTIVE SUMMARY

Summarized it can be said that the PieMenu is the clear-cut overall "winner" while the following usability problems have the highest severity and should be tackled first:

- Chameleon Touch-Pad button clicks must be made easier.
- Chameleon Touch-Pad slider should be used for relative highlight bar movement in the ListMenu.
- Chameleon Touch-Pad interpretation of slider and button click must be improved, concerning both lag and differentiation. Every contact to a single point for any period of time should be recognized as a click, when the finger is again lifted from this point. Whenever this single point of contact "moves", a yet to be determined distance, it should instead be recognized as slider movement.
- Individual sensitivity profiles should be considered for both the Chameleon Touch-Pad slider and the required Intersense InterTrax^2 rotation range
- Chameleon Touch-Pad attachment positions at thigh and belt area should be evaluated

12 Conclusion

A beginning has been made, but there is still more work to do.

In my opinion, usability engineering for Ubiquitous Computing is very exciting and deserves much more research than what is possible in one diploma thesis.

This chapter concludes my thesis. I will first present a summary of the outcome of my work. Next I will talk about my personal experiences during the last six months and finally share a few ideas on future work.

12.1 Results

The most important result is certainly the software framework to start conducting usability studies within DWARF. This includes a fully automated logging tool to capture events from the running DWARF system. It also brings a manual data entry tool to take quick written notes for later review. All performance measurements can finally be visualized live during the study or off-line for high quality prints with a number of highly flexible and adaptable scripts.

I have conducted a usability study, where two menu type setups have been compared which were not set against each other before, with a plethora of interesting results. By doing so I have shown that the usability evaluation framework proposed in this thesis works both on the process as well on the software level. Finally, this sample study has thrown off sample study materials which can be used as guidelines for future studies.

12.2 Lessons Learned

Conducting the sample usability study has been a very interesting experience. I was very positively surprised on how much data could be extracted even from mostly internal participants with only three least competent users added to the mix. About nine participants have shown to be a very good number to reveal the most prominent usability problems but to also gather many unexpected details. Planning and conducting a pilot study well ahead of the final one for last minute fixes to both the setup and the evaluation materials has proven to be mandatory.

It required a fair amount of discipline from me, the usability monitor, not to influence the participants prior or during the study in any subtle way.

It seemed advisable to log all possible data during the study, since you might notice late in the study unexpected problems, which require additional measurements not included before. There is no way to re-evaluate the already evaluated participants with the added measurements, so all raw data should be logged from the very start to keep all options for data mining.

The results of the study have convinced me not to trust the usability of existing designs unless they have been proven with profound trust-worthy usability evaluations or unless you have verified them yourself with at least a quick pilot study.

The study setup I developed and used required no funds, leveraged existing hardware to combine them in new human interface metaphors which were worthy enough to be evaluated, and was conducted by only one person, and yet still yielded noteworthy results. Looking back at the achievements with the very limited resources, I'm excited about what is possible with funds where new promising hardware can be bought or developed and combined in new ways, exploiting a much bigger design space aiming at making Wearable Computing and similar technologies available to everyone at optimal levels of usability.

The occupation with usability related topics for six months has shifted my personal focus in initial product reviews for the better. Whereas I earlier first studied e.g. offered functionality and technical details of a new product, I'm now much more critical about usability aspects, comparing it to the usability of similar products, while trying to reveal inconsistencies or usability problems with matching recommendations on how to improve on them.

Finally, working in the ARCHIE team on a common project has been a rewarding team work experience although it has introduced many dependencies which have delayed the sample study much more than I initially had anticipated. Avoiding these problems is part of future work.

12.3 Future Work

The topic of usability engineering for Ubiquitous Computing is very vast and can and should be covered by future work e.g. in form of new practical courses, master theses, and even doctoral theses. The following list gives an overview about possible follow-up topics.

12.3.1 Integrate Log4J Type Logging into `DataLogger`

The notion of logger hierarchies to achieve log statements at arbitrary granularity ([5], Chapter 8.1) should be integrated into `DataLogger` for much higher flexibility than what is currently possible with a flat file structure.

In the long run all data should be logged, eliminating all semantic mapping still existent in the current implementation (See chapter 10.1).

This creates more future work for the visualization tool, since this setup requires much more complex log data interpretation functions for post-semantic mapping.

12.3.2 WIMP paradigm for Augmented Reality

There is no established WIMP [60] type paradigm for Augmented Reality and similar technologies yet. By conducting many evaluations studies at a low level comparing simple interfaces for selection, de-selection, deletion, movement, multi-selection etc., a similar paradigm might be found. This would be a major step towards usability for these technologies.

12.3.3 Mobile Usability Monitor

Currently the usability monitor has to rely on an entry mask to type in observations and to control the study. This is fine when in a laboratory, with a computer next to the monitor. However, studies get really interesting when they are performed outside in the real-world, which takes away all standard GUI interfaces from the usability monitor forcing him to rely on Augmented Reality technologies to control the study as well. Speech recognition could for example be used to take notes. However, great care has to be taken to allow this happen without letting the participant hear the note being taken.

Real mobile studies also pose many new observation problems which are to be solved in future work. Ideally, the usability monitor would be on some type of automated mobile transportation device following the participant at an angle allowing good observation. This way he will not have to worry about keeping up with the participant and can concentrate on the action at hand. Alternatively, a suitable camera setup on the participant or in the environment should be found as already shortly discussed in chapter 8.

12.3.4 Object Oriented Visualization

In the long run the scripting approach for visualization is probably not feasible, although it has proven to be very effective for rapid prototyping. In the future an object oriented approach should be taken, allowing better levels of re-usability of previously written components.

12.3.5 Electronic Questionnaires

It takes quite some time to digitize all the written questionnaires so an electronic format for faster summarization should be developed. However, this solution must go through extensive usability studies itself since we do not want to lose subjective participant feedback because they cannot use the electronic questionnaires like the printed ones. I envision a flat tablet PC solution with a pen for hand writing recognition here.

Alternatively, the filled out questionnaires could, of course be scanned and processed by OCR (Optical Character Recognition), if it turns out that the paper forms always surpass the usability of any electronic ones.

12.3.6 Authoring Tool for Quick Conceptual Model Mock-Ups

There is no authoring tool for Augmented Reality applications to quickly create conceptual mock-up prototypes in a fast iterative fashion for usability evaluations. This tool would

greatly enhance the usability engineering process and should be written to assist both the development team and the usability monitor.

This tool should have two modes. One for the usability engineer to freely experiment beyond restrictions and one mode for developers which forces compliance to pre-set usability guidelines effectively becoming a CAUSE (Computer Aided Usability Engineering) tool for Augmented Reality.

12.3.7 Authoring Tool for Task Design

Currently the authoring of tasks for inclusion in automated task time and name counter is open for many improvements. Currently it is hard-coded with special services. An authoring tool should be developed e.g. on the basis of UICs in DWARF, which could allow quick, yet complex task generation for easy later reproduction.

12.3.8 Wizard of Oz Tool

For more complex studies it is unreasonable to wait until all functionality is implemented to start the first usability evaluations. A usable "Wizard of Oz [49]" tool should be developed which allows the usability monitor to simulate missing functionality with simple button clicks. An extension of the already existing ARCHIE product called `DISTARB` [66] would be well suited for this. Alternatively the Jfern [19] Petri net framework could be leveraged.

12.3.9 Measurement and Visualization Mapping

Much more research has to be put into measurement methodologies. How can we measure for example system lag to gauge nausea of the user when submerged in a not optimally responsive virtual or augmented world? Maybe even a good mapping can be found from measurement type to optimal visualization representation.

12.3.10 Improve `DataEntry` and `DataLogger` Integration

The integration between `DataEntry` and `DataLogger` is far from being exhaustive. `DataEntry` should be usable for example to stop all logging or to delete previously already saved persistent data when a task had to be reseted e.g. due to technical difficulties.

Timeouts should also be automatically detected by the logging system without requiring the usability monitor to watch this. This implies means to specify maximal allowed task completion times prior to the study.

12.3.11 Increase Visualization Tool Integration

Currently certain scripts for visualization have to be pre-selected prior to the study by the usability monitor which will then be constantly updated with live data.

However, sometimes it is not clear in advance which visualization is needed of which measurement type during a given study. To change this visualization on demand the usability monitor should not be bothered with going to a command line and type in complicated parameters. Instead he should e.g. only have to click on a map representing the system state and exchanged events similar to what e.g. DIVE [47] already offers to DWARF. By clicking on these communication relationships the corresponding visualization should come up.

I also see potential for smart visualization which always shows visualizations of measurements exceeding or falling below pre-set thresholds.

12.3.12 Cover more of the Usability Engineering Lifecycle for Ubiquitous Computing

Only a very small part of the full lifecycle, namely the usability evaluation, could be covered in this thesis. It is future work to cover the other phases and adapt them to Augmented Reality and similar technologies.

A Appendix

How is the visualization tool installed and used? The setup of the ARCHIE usability scenario demo and the sample study materials are also included.

A.1 Usage of Visualization Tool

A.1.1 Installation

The chosen visualization tool ploticus[1] has only been tested on i386 platforms running Linux but since its source code is available under the GNU General Public License, it should compile and run on other systems too.

Pre-compiled binaries/scripts for the above mentioned target platform can be installed by extracting the following prepared files into the *$HOME* directory of the usability evaluation monitors machine. Ploticus version 2.10 was used.

Refer to the edited *Makefile* included in the files below to get an idea on which changes are required to build a full featured ploticus binary from scratch using possibly updated source code.

ploticus-bin.tar.gz This file contains all required binaries. All possible output features have been enabled in this build. A *proc_processdata.c* fix enabling further post-processing of datasets which was posted by the author as an interim solution until the next version has been applied as well. Some of the custom added scripts make use of this feature.

ploticus-lib.tar.gz The computer cluster at the chair was missing the GD Graphics Library[2] required for advanced ploticus visualization. For convenience this file includes all missing libraries in a pre-compiled fashion.

ploticus-include.tar.gz Herein ploticus and GD Graphics Library specific include files are to be found.

[1] http://ploticus.sourceforge.net
[2] http://www.boutell.com/gd/

ploticus-production.tar.gz All custom ploticus scripts, TTF (True Type Fonts) and the prepared source tree is included in this file. To ease fresh compilation attempts the scripts have also been packaged in an extra file *ploticus-scripts.tar.gz* just like the true type fonts in *ploticus-ttf.tar.gz*.

ploticus-vim.tar.gz For convenience this file contains Vim[3] settings for syntax highlighting of ploticus scripts.

To conclude the installation the following section has to be added to the *.bashrc* in the *$HOME* directory:

```
# <ploticus settings>
PLOTICUS_PREFABS=$HOME/ploticus/plsrc210/prefabs
export PLOTICUS_PREFABS

if [ $HOSTTYPE = i386 ]; then
  export LD_PRELOAD=$HOME/lib/libgd.so.2
fi

GDFONTPATH=$HOME/ploticus/ttf
export GDFONTPATH
# </ploticus settings>
```

The above settings are to be adopted, if the chosen installation location departs from the above recommendations.

A.1.2 Scripts

Please refer to chapter 10.1 for a discussion on the data log file format to understand on what the script files are operating on.

For this thesis' experiments four custom ploticus scripts were put together.

absolutebars.plo This script needs the *Study*, *Task*, and *Type* name to filter by and will output bars showing the absolute totals of all different *Value* occurrences. If no *Participant* was specified, it will output average bars together with a specification on how many participants the script averaged over. If a *Participant* name is given, it will output bars using data from this specific user only.

The sample visualizations in figure A.1 show both the average bars for all five and for the special *Participant* "MartinB"in the "PieMenu-2"task. The bars show all *Value* outcomes for the "MenuSelection"*Type*.

[3] http://www.vim.org/

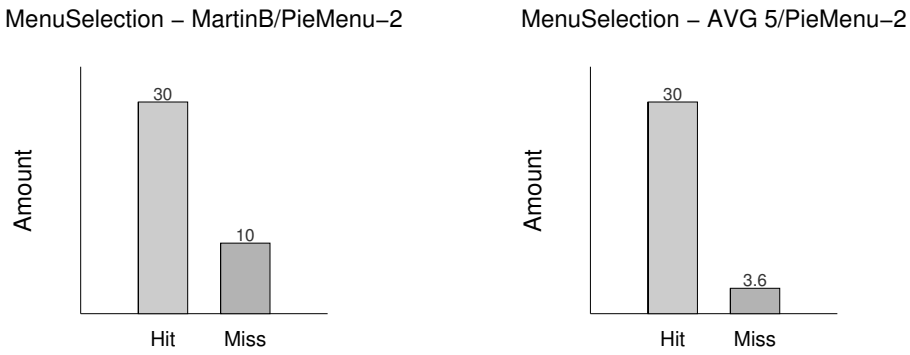


Figure A.1: Sample visualization, Left: standard absolute bars; Right: averaged absolute bars

absoluteerror.plo Here results are filtered by *Study*, *Task*, *Participant*, and *Type*. The y-axis shows all possible *Value* occurrences, while the x-axis shows time values. The dots show when each value was logged.

The sample visualization in figure A.2 shows when a *Value* of ”Hit”or ”Miss”was logged in the ”MenuSelection”*Type* category for the *Participant* ”MartinB”. Again task ”PieMenu-2”was used to filter by.

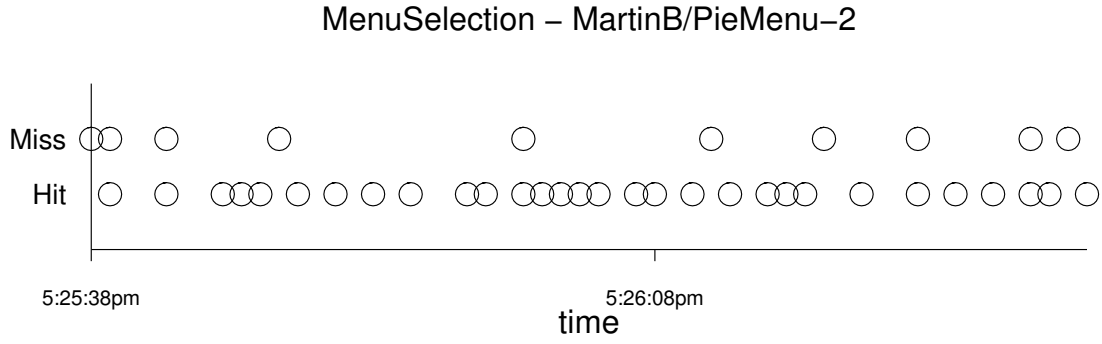


Figure A.2: Sample visualization, absolute error

relativeerror.plo Having the exact same parameter requirements as the former script, the matching hit-ratio development over time is visualized depicting the relative error (Figure A.3). This script is the least flexible of all four, since it requires the *Values* to be exactly ”Hit”or ”Miss”to calculate the relative error. A box in the top right corner always shows the final hit ratio value.

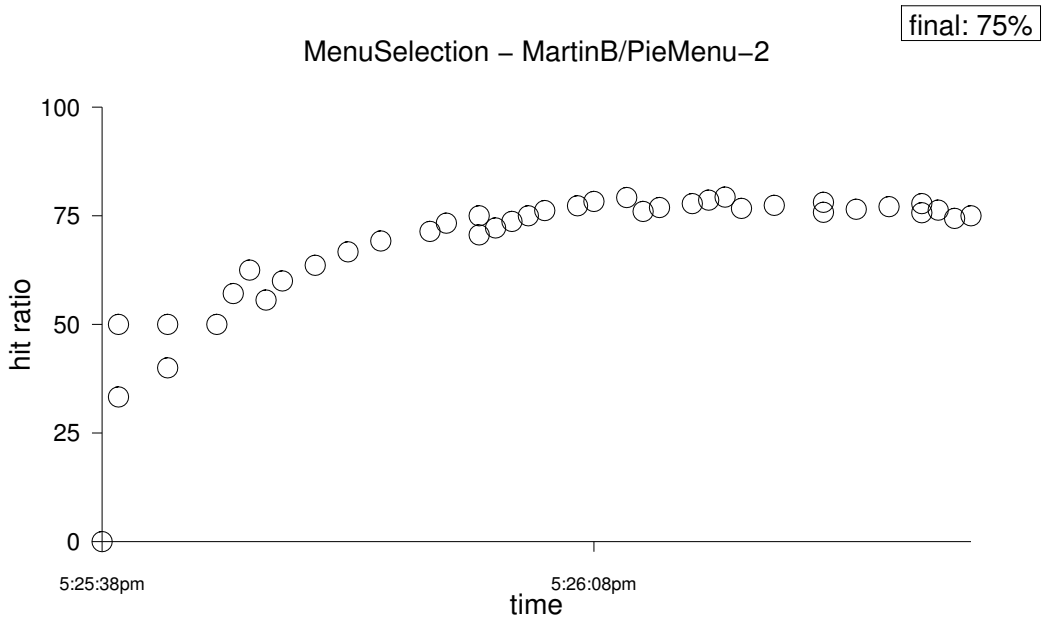


Figure A.3: Sample visualization, relative error

timerange.plo This script requires merely the *Study* name and will then show a task time completion range distribution in form of an error range bar for each task (Figure A.4). The y-axis is depicting the task completion times whereas the x-axis provides a listing of all performed tasks in this study. Below the task names a number is given, which states the number of *Participants* who took part in it.

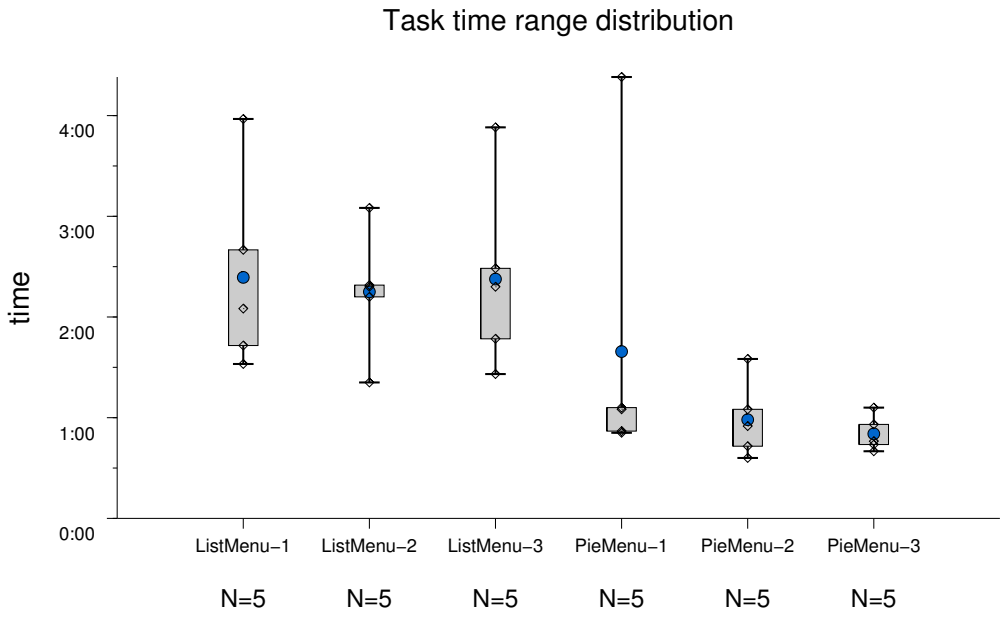


Figure A.4: Sample visualization, time range script

Per default the error tails extend to the minimal and maximal values while the range-box borders are set at the 25th and 75th percentile. The ”light”circles reveal all individual task time completion values, whereas the ”blue”circle is placed at their average value.

The range-bar drawing behavior can easily be changed however depending on the current requirements.

Parameter passing and interactive display is all handled by a custom wrapper script *pl_wrapper* with the following command line syntax:

```
pl_wrapper v2.0c (c)2003 kulas@in.tum.de

pl_wrapper -d datafile [-f outputformat -w winloc -o filename]
[-s study [-p participant] [-k task]] [-t type] scriptfile
  datafile      - input datafile with log data
  output format - non i-x11 (interactive) pl output format
                  (use x11 for standard non-updating display)
                  (use paper for high quality paper display)
  winloc        - sets x11 window location (x,y) for x11 output modi
  filename      - filename for non x11 pl output
```

```
study         - filter results by this study (non i-x11)
particpant    - filter results by this participant (non i-x11)
task          - filter results by this task (non i-x11)
type          - filter results by type
scriptfile    - ploticus scriptfile
```

datafile No matter which scriptfile is to be executed, the input file with the log data has to be specified at all times.

outputformat The default outputformat is *i-x11*, which is the custom interactive x11 display. This mode is meant for live visualization during an usability study. The wrapper checks the input file every three seconds (customizable) for changes and updates its visualization, whenever a change has occurred.

In the interactive x11 display, all parameters *Study*, *Participant*, and *Task* are disregarded, since the wrapper will take the current live values out of the last added log line at each update anyway for a truly live display without intervention of the usability evaluation monitor.

If "x11"is chosen here, a standard non-updating x11 window is rendered with none of the above mentioned live features.

"Paper"is another special parameter, which will render a high quality true type document in *eps* format and display it on the screen using ghostview[4] immediately.

Other formats available are:

```
ps eps svg svgz png jpeg wbmp FreeType2
```

winloc When either the *i-x11* or standard *x11* output format was chosen the window location can be specified here in (x,y) coordinates of the display.

filename For non "x11"-type output, a file is generated which is named "output"per default. By passing this parameter, a custom name can be entered.

study This parameter specifies the *Study* name, by which to filter in the script.

participant This parameter specifies the *Participant* name, by which to filter in the script.

task This parameter specifies the *Task* name, by which to filter in the script.

type The *Type* to filter by is required for every custom script but "timerange.plo".

[4]http://www.cs.wisc.edu/ ghost/

scriptfile The final mandatory parameter declares the desired custom ploticus script file which should be executed.

The interactive *i-x11* display can be e.g. leveraged to live update all four visualizations at once (Figure A.5) on the computer desktop.

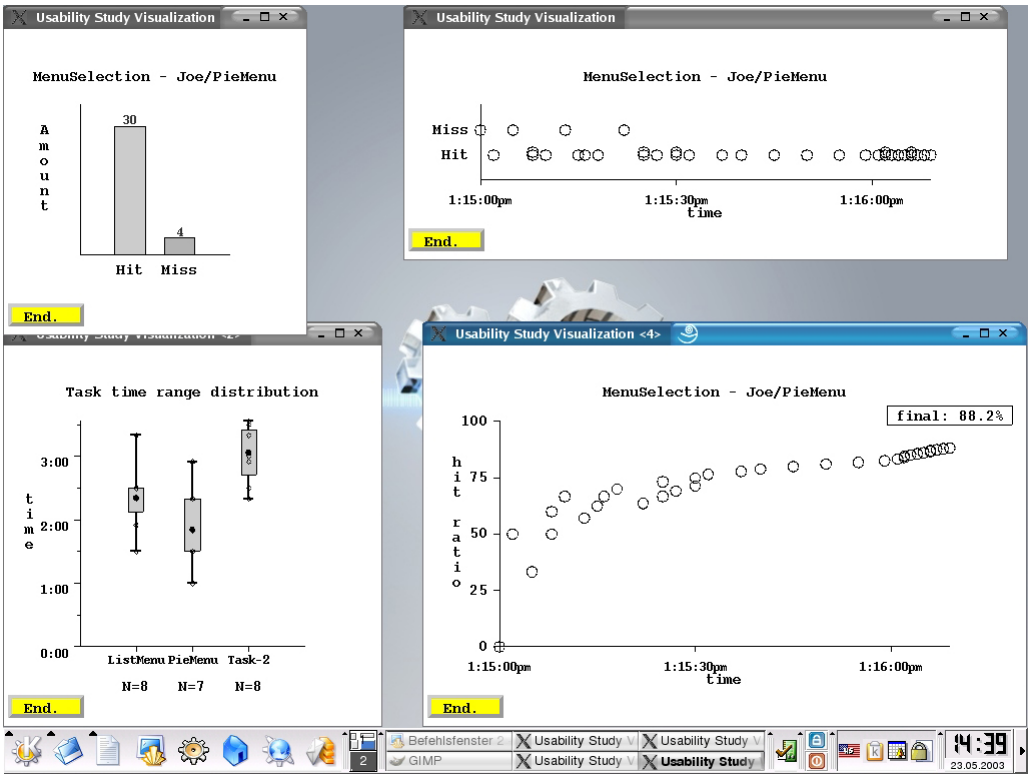


Figure A.5: Sample visualization, live interactive overview

Two simple scripts such as

```
file: run-iOverview
#!/bin/bash

# NW
(./pl_wrapper -d $1 -w 0,0 -t $2 absolutebars.plo >/dev/null)&
# SW
(./pl_wrapper -d $1 -w 0,500 timerange.plo >/dev/null)&
# NE
(./pl_wrapper -d $1 -w 400,0 -t $2 absoluteerror.plo >/dev/null)&
# SW
(./pl_wrapper -d $1 -w 800,500 -t $2 relativeerror.plo >/dev/null)&

file: kill-iOverview
#!/bin/bash
```

```
killall -9 pl_wrapper pl sleep run-iOverview
```

accomplish this task, by rendering each x11 window at a different location on screen. Each *pl_wrapper* call updates its own view in the background automatically.

The interactive overview is then started by passing the parameters for the *Datafile* and *Type* to the above mentioned script in a call like

```
./run-iOverview "/tmp/study2.log" "MenuSelection"
```

A.2 ARCHIE Usability Scenario Demo Setup

Here the correct start order of services to repeat the ARCHIE demo, is described.

Please refer to Section A.3.3 for details on how to start all the usability related services. Additionally to these the following services have to be started:

VideoGrabber This service [30] grabs video data from a FireWire camera leveraged by the *MarkerDetection* service which needs visual data.

MarkerDetection *MarkerDetection* detects markers which are e.g. placed at the hallyway walls / doors and outputs the current room information [37] to *DummyMenuList-Provider* which was modified for the demo to update the menu according to the current room.

MarkerConfiguration This service is complementary to *MarkerDetection* [37].

MenuToSpeech *MenuToSpeech* receives "MenuSelection" events from the PieMenu and notifies *derVorleser* about which text to read aloud

derVorleser This service plays audio files for the demo.

A.3 Sample Study Materials

Here all the materials which were used for the sample usability study are to be found.

A.3.1 The Orientation Script

"Hi, my name is Chris Kulas. Thank you for attending this usability study today. All of this will take about 30 minutes of your time.

We are here to test-drive a new interface paradigm. You will be asked to perform menu-selections with multiple new input devices you will be wearing together with a head mounted display. I will be here monitoring and logging various data to later judge which menu system is better suited for our purposes.

The menu system is tested, not you as a person, so don't feel pressured in any way. If something unexpected happens, it's always the computers fault, not yours. Perform the tasks like your normally would in a calm way. You are not competing against anyone, so do not rush. Please think aloud if any thoughts come up.

You will be asked to fill out multiple forms and answer other questions I might have for you. Please tell me honestly what you really think and don't tell me what you think I might want to hear.

Thank you.

Do you have any remaining questions? Feel free to ask anytime during the tasks when something comes up."

A.3.2 General Usage Guidelines

General menu usage After attaching the input devices, we will put on the head-mounted display and adjust it for a comfortable fit. In the display you will see a menu centered on the screen.

For each menu type you will be asked to perform three identical tasks each consisting of 30 menu selections. At first you will just see one entry labeled “Start Task!”. Feel free to start the first task by pressing the button at any time when you feel ready. During each task you advance by selecting the “Select Me!” entries, until the menu shows only one big entry reading “Completed! Restart?”. Selecting the “Disabled” entries will do nothing.

When you hit “Completed! Restart?” the prior task will repeat itself. Please do so two times for a total of three runs. You may rest between each task if you want.

You may only use one hand to navigate the menu. You are free to put your other hand wherever you please as long as you don’t touch your primary arm with it. You should stand upright during each task at all times. While using the menu you should stand on the marked spot and face the white wall.

One of the input devices you will use in both setups is the Chameleon Touch-Pad. It is a giant button. You can press the button by tipping it with the tip of one of your fingers for a short moment. The shorter you press this “button”, the better. You always press this button when you want to select any menu entry.

ListMenu usage In the ListMenu (Figure A.6), items are aligned vertically.

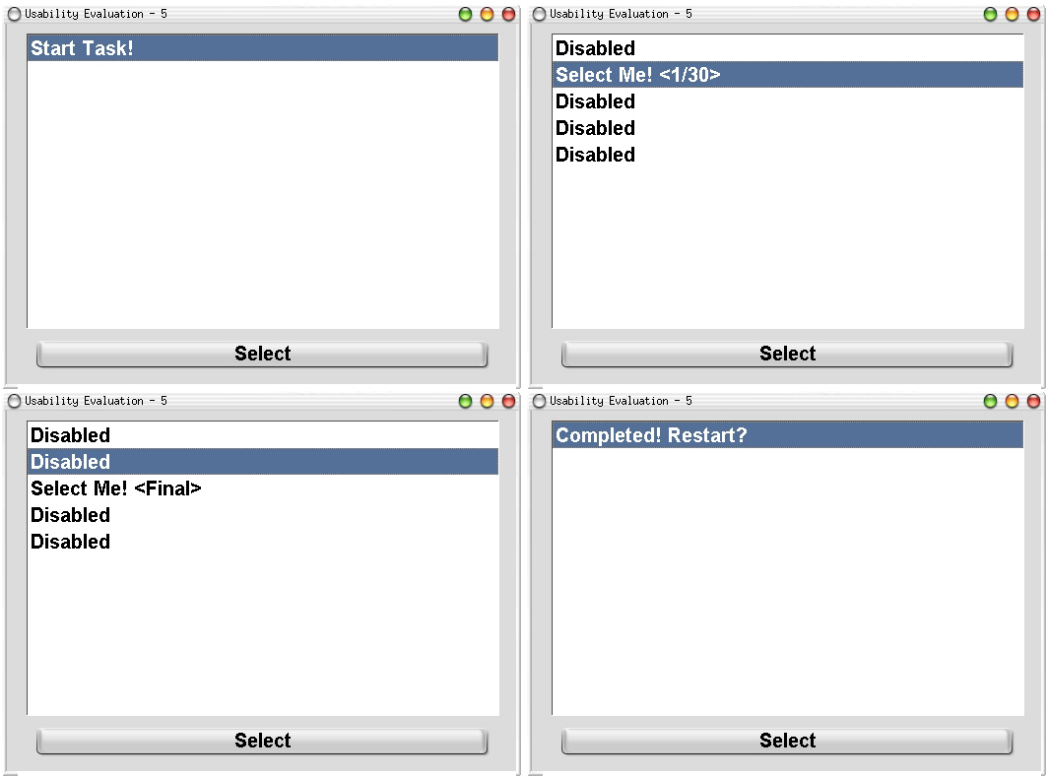


Figure A.6: ListMenu-Task

In this mode the Chameleon Touch-Pad (Figure A.7) is your only input device. We will attach it to your left breast area using a special backpack with velcro on it.

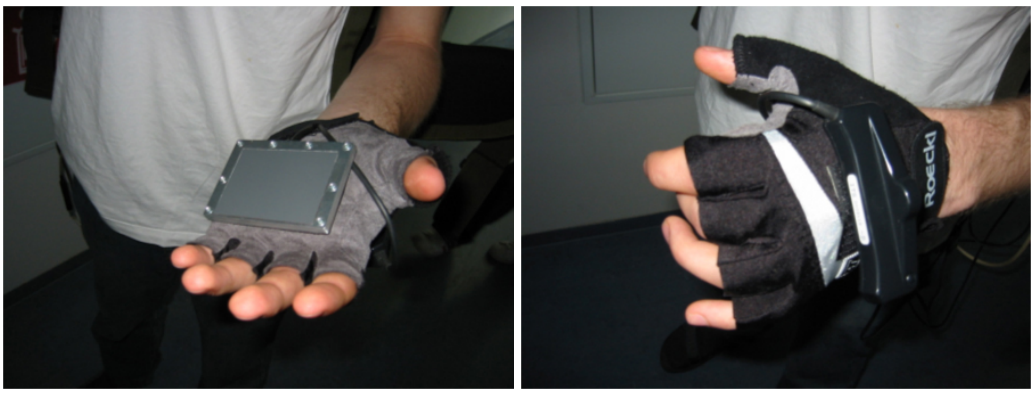


Figure A.7: Left: Chameleon Touch-Pad; Right: Gyroscope

Here the Chameleon Touch-Pad isn't just only a button but also a slider. You can use one of

your fingers on your right hand to slide along on the Chameleon Touch-Pad to highlight the corresponding menu entry. Sliding right will highlight items further down, sliding left will highlight an item further up. You may attach the Chameleon Touch-Pad differently though if you wish. After you have highlighted the item you want just remove your finger straight away and press the button like you normally would by tipping the Chameleon Touch-Pad quickly for selection. Sliding works best when you keep your finger on the Chameleon Touch-Pad for a longer period and when you move your finger slowly. When you slide, it is best to minimize the area of contact with the Chameleon Touch-Pad so try to use only the tip of one of your fingers. If the slider does not seem to work, remove your hand from the surface and try again.

PieMenu usage In PieMenu mode, items are arranged on an upper half-circle (Figure A.8).

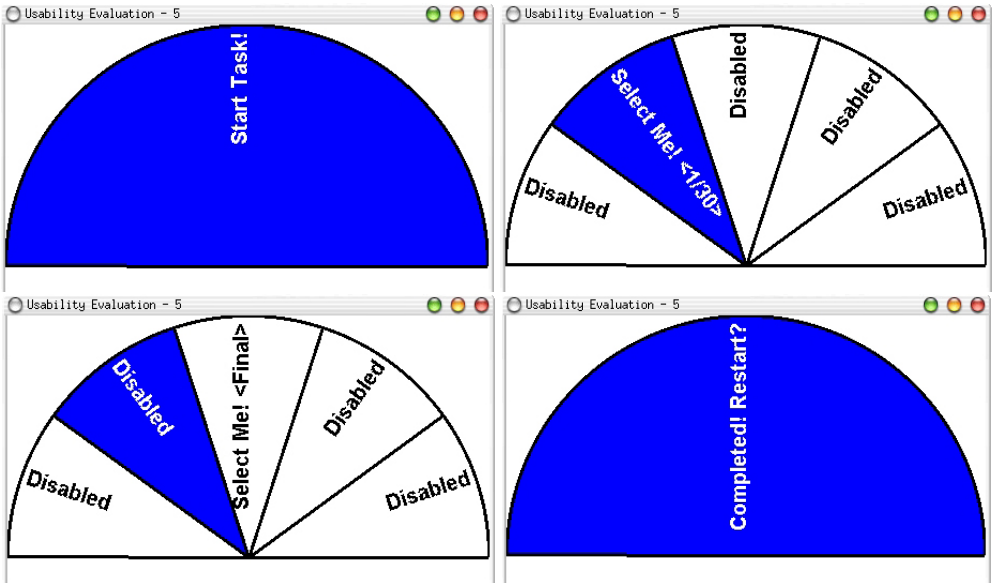


Figure A.8: PieMenu-Task

In this mode you will be wearing a glove with two input devices attached to it. The Chameleon Touch-Pad will be fixed to the palm portion of the glove whereas a gyroscope will be attached on the top side just like in figure A.7.

The Chameleon Touch-Pad is here really just a simple big button for menu selection. The gyroscope will register your hand rotation on the z-axis for entry highlighting.

When you rotate your hand to the left (counter-clockwise) you will highlight an item further to the left in the half circle, just like rotating your hand to your right (clockwise) will highlight an item further to the right. You should never have to rotate your hand more than about 45 degrees to your left or 135 degrees to your right (Figure A.9).

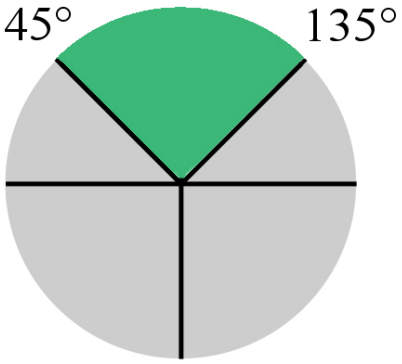


Figure A.9: Hand rotation range for the PieMenu

So a rotation of about 90 degrees is sufficient to go from the entry on the left-most side to the entry on the right-most side or vice versa. Highlighting works best when you rotate your hand slowly in a calm way, while pointing straight ahead with it.

When you want to rotate your hand in another "section" (green) of the full circle (grey) just rotate your hand way over to the left to rotate the green pie counter-clockwise or rotate it way over to the right to move it clockwise. Going slowly back will immediately highlight the item next to the last item on the edge.

After you have highlighted the item you want just press the button like you normally would while keeping your hand in a fixed position.

A.3.3 Miscellaneous

Usability Evaluation Task Menu Entry Selection Distribution

The 30 randomly but fix picked numbers for the selection sequence in the tasks resulted in the following open entries in a five entry encompassing menu:

```
2 1 5 3 5 2 3 1 2 5 4 1 5 3 2 3 4 4 1 5 3 2 3 4 5 1 2 5 2 3
```

which results in the following per entry frequency:

1. five times
2. seven times
3. seven times
4. four times
5. seven times

Service Start Order

Here the correct start order of services to conduct the sample study. If everything was correctly executed DIVE should look like figure 11.10 on page 124 for the ListMenu or like figure 11.9 on page 120 for the PieMenu.

```
kulas@atbruegge34 [Configuration]
1. ./run-servicemgr
2. ./Configuration

demo@atbruegge10 [Hardware + Logging]
3. ./run-servicemgr
4 ./IntersenseTracker (if PieMenuTask)
        check: "InterTrax detected on port 1"
5. ./TouchGloveService -DserialPort=/dev/ttyS1
        check: "INFO:Read mode byte: 8B."
8. ./DataLogger -Dfile=/home/kulas/session.log -Dcounter=y
        check: NO PieMenuDisplay.xml
               NO ListMenuDisplayStatic.xml
9. ./DataEntry

kulas@lapbruegge49 [Menu View with Glasstron]
        check: NO DataLogger.xml
        check: NO DataEntry.xml
        check: NO TouchPadGlove*.xml
6. ./run-serviemgr
7. ./PieMenu
        or
   ./ListMenu -Dusability=yes
        -> runOnDemand: TouchGloveInterpreterService (ListMenu, PieMenu)
                        MUST RUN on machine were TouchPadGlove is attached
        -> runOnDemand: PoseDataAngleInterpreter (PieMenu)
10. ./DummyMenuListProvider
```

A.4 Summarized Sample Study Questionnaires

This section contains all the summarized sample study questionnaires. Similar answers in the free form questions were grouped together, where the number indicates the frequency. *Cursive text indicates comments by me, the usability monitor*

A.4.1 Background and Pretest Questionnaire

Please circle your answer.

Demographics

1. You age falls into which category?
   a) 18-29 (9)
   b) 30-39
   c) 40-49
   d) 50+
2. What is your sex?
   a) male (7)
   b) female (2)
3. Are you left or right handed?
   a) left-handed (1)
   b) right-handed (7)
4. What is your highest level of education?
   a) Hauptschulabschluss
   b) Abitur (1)
   c) FH student majoring in: Social Pedagogy (1)
   d) FH with major:
   e) University student majoring in: Bachelor CS (1), Master CS (3)
   f) University with major: Master CS (2), ? (1)

   This question caused some confusion. Often people circled their last completed degree although they were currently majoring in something right now.

Computer experience

1. What is the total length of time you have been using personal computers?

   1/2 year, 3 years, 5 years, 9 years (2), 12 years, 13 years, 15 years, 22 years, (= *AVG 9.83 years*)

2. For how many hours do you use a computer on a typical work day?

   1 hour, 2 hours (2), 6 hours, 8 hours (3), 9 hours, 10 hours (= *AVG 6 hours*)

3. How does your computer interface usage add up to 100% ?

0 (3), 9, 20, 30, 40, 70 (2) (= *AVG 26.5*)	%	Command Line
20, 25, 60, 68, 80, 90, 100 (3) (= *AVG 71.4*)	%	Graphical User Interfaces
0 (5), 1, 2, 5, 10 (= *AVG 2*)	%	Augmented Reality or Wearable or Tangible Interfaces
100	%	All of the above

   Skip ahead to the next section if you specified 0% for Augmented Reality interface usage.

4. What is the total length of time you have been using or testing Augmented Reality type products?

   0 (5), 2 hours, 1 year (50-100h), 3 years (2)

5. For how many hours do you perform tasks using Augmented Reality on a typical work day?

   0 (8), 0.5 hours

Initial impression

1. The ListMenu task seems easy to perform.

   a) Strongly Disagree
   b) Disagree (1)
   c) Neither Agree nor Disagree (2)
   d) Agree (4)
   e) Strongly Agree (2)

2. The PieMenu task seems easy to perform.

   a) Strongly Disagree
   b) Disagree (1)
   c) Neither Agree nor Disagree (3)
   d) Agree (4)
   e) Strongly Agree (1)

Preferences

1. What is your learning style preference?

   a) Read then do
   b) Try then read (3.5)
   c) Learn by doing (5.5)

   one user selected both here with text "depends on my needs"

2. In general, I enjoy using high-tech products.

   a) Strongly Disagree (1)
   b) Disagree
   c) Neither Agree nor Disagree (1)
   d) Agree (4)
   e) Strongly Agree (3)

A.4.2 Posttest Questionnaire - ListMenu

Please circle your answer. Fill out once for each menu type.

Readability

1. The menu was placed at a good readable location on screen.

   a) Strongly Disagree
   b) Disagree
   c) Neither Agree nor Disagree
   d) Agree (4)
   e) Strongly Agree (5)

   Right in the middle of my sight.
   A little more upwards would have been better.

2. Menu entry readability was satisfactory.

   a) Strongly Disagree
   b) Disagree (1)
   c) Neither Agree nor Disagree (3)
   d) Agree (3)
   e) Strongly Agree (2)

Font was too small (3).
Better readable than PieMenu (because letters were leveled).
Would have been sharper if I had my glasses on.

3. It was always clear which menu entry was currently highlighted.

   a) Strongly Disagree
   b) Disagree
   c) Neither Agree nor Disagree (1)
   d) Agree (3)
   e) Strongly Agree (5)

   There's a highlight-bar.
   Yes, but PieMenu was better recognizable, because Oka-Yellow worse in HMD (*Wrong color in ListMenu Bar for ALL Tasks by accident, still low impact*).

Usability

1. Highlighting the desired item was easy.

   a) Strongly Disagree (2)
   b) Disagree (2)
   c) Neither Agree nor Disagree (1)
   d) Agree (3)
   e) Strongly Agree (1)

   Needed a little practice.
   But sometimes there were ambiguous hops.
   Especially with the right hand I had difficulties (left handed) (*This user tried the ListMenu in once task with the left hand, no measurable improvement*).
   Sometimes it was good, sometimes no reaction (probably because the difference of "Button-Click" and "Slide" were not comprehend-able to me).
   When releasing the finger the menu jumps up or down, but mostly up.
   Very imprecise control of highlight-cursor.

2. Selecting the already highlighted item was easy.

   a) Strongly Disagree
   b) Disagree (1)
   c) Neither Agree nor Disagree (2)
   d) Agree (5)
   e) Strongly Agree (1)

The finger can remain at the same position (*Misunderstanding, corrected from Strongly Agree to Agree, thought pre-selected. Used Double-Clicks!*).
Sometimes the selected item seemed to move one item above by "clicking".
See Above (*couldn't differentiate click and slide*).
A few times I wanted to select, but moved the cursor to another item.

3. Each item in the menu was equally easy to highlight and select.

   a) Strongly Disagree (1)
   b) Disagree (3)
   c) Neither Agree nor Disagree (1)
   d) Agree (2)
   e) Strongly Agree (2)

   Seemed as if the edge items would be a little more difficult to select (*leveraged direct mapping later*).
   But you have to get familiar with the UI.
   Top and down are easier.
   Equally hard ☺.
   The bottom one was a little hard to highlight.
   Last was the most difficult.
   The lower ones where harder to select.
   Top-most and bottom-most were easier.

4. The required range of motion on the device for highlighting all entries was adequate.

   a) Strongly Disagree
   b) Disagree (2)
   c) Neither Agree nor Disagree (2)
   d) Agree (3)
   e) Strongly Agree (2)

   See ambiguous hops on earlier question.
   Sometimes the highlight bar seemed to move fast, sometimes too slow.
   For me (as a laptop user) it was unfamiliar to me that the highlight was not relative to the movement, but actually depended on the absolute position of the finger (*exploited direct mapping later in third run, with no noteworthy time win though*).
   It could be a litter faster (*distance too big, own mouse also set to a very fast speed*)
   Breasts were of hindrance (*female participant*)

Consistency

1. The menu always behaved like I expected it would.

a) Strongly Disagree
b) Disagree (4)
c) Neither Agree nor Disagree (2)
d) Agree (1)
e) Strongly Agree (2)

Sometimes a little "lag".
See Above. (*unfamiliar touch-pad behavior*)
Not that easy to handle.
See usability comments.

2. The menu behaved the same during all repeated tasks.

a) Strongly Disagree
b) Disagree (2)
c) Neither Agree nor Disagree (1)
d) Agree (4)
e) Strongly Agree (2)

Seemed to move in different speeds.
Hmm... towards the end the highlighting and selecting was better understood and I had less trouble.

Physical comfort

1. I felt physical discomfort while using the menu.

a) Strongly Disagree
b) Disagree (3)
c) Neither Agree nor Disagree (4)
d) Agree (2)
e) Strongly Agree

The pad position was not that comfortable, the glasses were excellent.
The active (right) arm is in a bad (=tiring) position for longer work, but for single actions: good!.
Having the hand at the breast all the time is a bit strange, esp. think of girls.
Well ☺ For a quick selection the PieMenu is better (the movement of the hand to the breast takes also time!).
Uncomfortable: HMD.

2. The menu would be easier to use if I was allowed to use both hands.

a) Strongly Disagree (3)
b) Disagree (4)
c) Neither Agree nor Disagree (1)
d) Agree
e) Strongly Agree

When writing, I only use one (the more familiar) hand...
Why?
No answer from one participant
One hand is enough ☺

3. The menu would be easier to use if I was allowed to attach the devices somewhere else on my body.

a) Strongly Disagree (1)
b) Disagree
c) Neither Agree nor Disagree (2)
d) Agree (3)
e) Strongly Agree (2)

Thigh.
When familiar with the UI, I can even use it on the backside of my head (*Didn't mean it would be easier, he simple meant he could use it anywhere*)
I'd suggest attaching it at the belt.
Well I guess that depend on what the actual user task is. Being able to place it individually rocks ☺
No answer from one participant
When using long - uncomfortable, on the arm probably more comfortable
Maybe better at waist. Rotate 90 degrees.

Conclusion

1. Overall the device and menu layout was easy to use.

a) Strongly Disagree (1)
b) Disagree (3)
c) Neither Agree nor Disagree
d) Agree (3)
e) Strongly Agree (1)

It's easy to understand.
I think I would have done to many "click errors".
This laaags so bad.. 1. reaction time shifted 2. no reaction (probably because I didn't

interpret highlight and select correctly).
No answer from one participant
Misunderstanding, Relative Highlight Bar Moving expected!

A.4.3 Posttest Questionnaire - PieMenu

Please circle your answer. Fill out once for each menu type.

Readability

1. The menu was placed at a good readable location on screen.

   a) Strongly Disagree
   b) Disagree
   c) Neither Agree nor Disagree
   d) Agree (3)
   e) Strongly Agree (6)

   I saw everything.
   The HMD slowly slips off the head, so the menu could leave the fov.

2. Menu entry readability was satisfactory.

   a) Strongly Disagree
   b) Disagree
   c) Neither Agree nor Disagree (4)
   d) Agree (3)
   e) Strongly Agree (2)

   Font to small (2), but I knew where to click.
   Difficult to read when the text is placed vertically, but I went after the visual impression anyway ("click me" is longer than "disabled").
   I don't remember ☺ (*Didn't look at readability, recognized entries by patterns*).
   Should have had glasses.

3. It was always clear which menu entry was currently highlighted.

   a) Strongly Disagree
   b) Disagree
   c) Neither Agree nor Disagree
   d) Agree
   e) Strongly Agree (9)

Definitely.
Color contrast was very fine.

Usability

1. Highlighting the desired item was easy.

   a) Strongly Disagree
   b) Disagree
   c) Neither Agree nor Disagree (1)
   d) Agree (4)
   e) Strongly Agree (4)

   Sometimes the gyro did not action.
   Most of the time, except when going from the left most entry to the right most. (*user wrenched his arm in study a couple of times*)
   Due to some system latency for a few seconds it was bad (lay, inertial tracker did not react), in general really good!

2. Selecting the already highlighted item was easy.

   a) Strongly Disagree (2)
   b) Disagree (1)
   c) Neither Agree nor Disagree (2)
   d) Agree (1)
   e) Strongly Agree (3)

   Everything good (*Misunderstanding: thought already pre-selected*).
   At the beginning of test, I had difficulties, because I moved the hand while pressing the "Button".
   The short click is difficult, sometimes I needed several attempts.
   With some experience ☺.
   Clicks took often several tries.
   Touch-Pad build for big hands. (*she had small hands, had to hold pad down with small finger to use it with index finger until she noticed she can also click at the top*)

3. Each item in the menu was equally easy to highlight and select.

   a) Strongly Disagree (1)
   b) Disagree (2)
   c) Neither Agree nor Disagree (1)
   d) Agree (2)

e) Strongly Agree (3)

The middle section was more difficult.
The leftmost sector was more difficult to select (I'm left handed).
Highlight: no, Select: Yes.
Middle was easier.

4. The required range of motion on the device for highlighting all entries was adequate.

   a) Strongly Disagree
   b) Disagree (1)
   c) Neither Agree nor Disagree (3)
   d) Agree (3)
   e) Strongly Agree (2)

   A bit less range for all items, or a bit too much range for my elbow-joint (*moved out of arm not using wrist!*)
   A little less would be nice, it's like being in the fitness studio otherwise (*didn't want to recalibrate, too troublesome, had suggestion: hold still, move back to middle*)
   I would make the range a bit smaller (*user also had mouse configured like this at home*)
   See previous answer (*Middle easier, wrenched arm a bit*)

Consistency

1. The menu always behaved like I expected it would.

   a) Strongly Disagree
   b) Disagree (1)
   c) Neither Agree nor Disagree (1)
   d) Agree (7)
   e) Strongly Agree

   Except when turning arm no corresponding highlight (*noticed drop outs*)
   Almost 90% of the time - yes!
   At first I tried to slide my hand. The menu had a mirrored behavior. (*!!*)
   Clicks not recognized.

2. The menu behaved the same during all repeated tasks.

   a) Strongly Disagree
   b) Disagree (2)
   c) Neither Agree nor Disagree (1)
   d) Agree (4)

e) Strongly Agree (2)

Sometimes behaved faster (*impression of less lag*)
Because of Learning I could improve velocity and Precision.
Almost at task ca. 80-84 there were lags (*end of third run*)
It bucked in the end (*lag*).

Physical comfort

1. I felt physical discomfort while using the menu.

   a) Strongly Disagree (3)
   b) Disagree (3)
   c) Neither Agree nor Disagree (1)
   d) Agree (2)
   e) Strongly Agree

   I could hold my arm in a relative normal position (*had feeling of being better tracked, when holding arm up*).
   Nerd and stuff.
   HMD only, Glove was o.k.

2. The menu would be easier to use if I was allowed to use both hands.

   a) Strongly Disagree (3)
   b) Disagree (3)
   c) Neither Agree nor Disagree
   d) Agree (1)
   e) Strongly Agree (2)

   Use other arm to support rotating arm.
   What about turn & speak (*instead of button*).
   Like driving car would be better.
   Problem: too short fingers.

3. The menu would be easier to use if I was allowed to attach the devices somewhere else on my body.

   a) Strongly Disagree (4)
   b) Disagree (4)
   c) Neither Agree nor Disagree (1)
   d) Agree
   e) Strongly Agree

Placing is good, eventually left (second) arm.
No idea without trying it...
Can't think of anything better spontaneously.

Conclusion

1. Overall the device and menu layout was easy to use.

   a) Strongly Disagree
   b) Disagree
   c) Neither Agree nor Disagree (1)
   d) Agree (2)
   e) Strongly Agree (6)

   Fine.
   Cool device!
   Only Problem: Click recognition.
   Hand not fixed when clicking on the leftmost entry, left-handed, use entry which was highlighted a milli-second earlier.
   What about pointer in PieMenu? like gas gauge?
   Third run was much faster since participant noticed that she can also click at the top of the pad, small hands!

A.4.4 Overall final questions

Please circle your answer.

1. I think for Augmented Reality menu selection this type of menu is best suited:

   a) ListMenu
   b) PieMenu (9)

   One has disjunct actions on different "body parts" (finger, arm vs. finger, finger).
   Provides optimum usage.
   I felt much more comfortable in my physical position, I can freely put my hand everywhere, less errors, very intuitive.
   Because of Multi-Modality.
   But the "Scotty-Beam-Me-Up" ListMenu is funny, too ☺. But not usable!
   Easier to use; natural movements

2. I was confused by the questionnaires.

   a) Strongly Disagree (2)

b) Disagree (5)

c) Neither Agree nor Disagree (1)

d) Agree (1)

e) Strongly Agree

The question whether the behavior of the menu had changed.
After: Usability 2 I awaited another Question on selecting a new item was easy...
OK

3. I would like to add the following:

I enjoyed the PieMenu
Point & click → ray-picking...
Nothing.

Glossary

API. *see* APPLICATION PROGRAMMER'S INTERFACE

AR. *see* AUGMENTED REALITY

Application Programmer's Interface. "Set of fully specified operations provided by a subsystem" [15]

Augmented Reality. A technique that uses virtual objects to enhance the user's perception of the real world.

Bluetooth. Standard for low range wireless communication.

CORBA. Common Object Request Broker Architecture. CORBA is a specification for a system whose objects are distributed across different platforms. The implementation and location of each object are hidden from the client requesting the service.

Data Glove. A glove equipped with sensors that sense the movements of the hand and interfaces those movements with a computer. Data gloves are commonly used in virtual reality environments where the user sees an image of the data glove and can manipulate the movements of the virtual environment using the glove.

DOF. Degrees of Freedom. Often specified to show capabilities of motion tracking devices.

DV. Digital Video. Video format e.g. used on DVDs for high quality recordings.

FireWire. *see* IEEE 1394

GNU. GNU's Not Unix. An initiative to re-implement the most common UNIX tools on an open source basis.

GPL. GNU Public License. A software license developed by the GNU initiative that allows redistribution of software in source code but restricts the rights of the licensee to commercialize software derived from the original code.

GPS. *see* GLOBAL POSITIONING SYSTEM

Global Positioning System. A wide range tracking system based on timing signals from satellites.

GUI. Graphical User Interface. *see* WIMP

Hallway Testing. Acquiring participants, who walk by in the hallway, for usability evaluations.

HMD. *see* HEAD MOUNTED DISPLAY

Head Mounted Display. A display device similar to glasses. Its user either sees only the display or the display information projected optically onto the real world (See-Through Head Mounted Display)

IDL. *see* INTERFACE DEFINITION LANGUAGE

IEEE 1394. Specification of a high speed (400 MBit/s) serial interface used to connect storage facilities or video cameras to computers. Branded *FireWire* by Apple and *iLink* by Sony.

Interface Definition Language. A language that allows the programming language independent specification of software interfaces. It is used to describe the interfaces of CORBA objects.

NTSC. National Television Standards Committee. American TV standard using 525 horizontal lines.

MVC. Model–View-Controller. MVC is the name of a methodology or design pattern for successfully and efficiently relating the user interface to underlying data models.

Middleware. Middleware is a general term for any programming that serves to ”glue together”or mediate between two separate and often already existing programs.

ORB. Object Request Broker. An ORB is at the heart of a CORBA system. Every process communicating via CORBA must have its own ORB running.

RAD. Requirements Analysis Document. A document describing the requirements of a software project and the way they were derived.

SDD. System Design Document. A document describing the general design of a software system. It serves as a basis for the implementation.

STHMD. *see* HEAD MOUNTED DISPLAY

TTF True Type Fonts.

Tracker. A device determining the position and orientation of a tracked object.

TUI. Tangible User Interface.

UML. *see* UNIFIED MODELING LANGUAGE

USB. *see* UNIVERSAL SERIAL BUS

Unified Modeling Language. ”A standard set of notations for representing models.” [15]. See [57] for details.

Universal Serial Bus. A convenient medium speed (12 MBit/s) serial interface available at most modern computers.

Use Case Diagram. UML notation to represent the functionality of a system.

VGA. Video Graphics Array. A PC display format specifying the minimum resolution and the number of colors which are to be displayed.

VR. *see* VIRTUAL REALITY

VRML. Virtual Reality Markup Language. Allows the convenient description of virtual objects and scenes for AR and VR applications.

Virtual Reality. A computer based technology that allows its user to act in purely virtual environments.

WaveLAN. A midrange wireless communication standard used to replace common Ethernet connections.

WIMP Windows, Icons, Mouse, and Pull-down menu. Common paradigm of GUIs [60].

XML. Extensible Markup Language. XML is a simple, standard way to delimit text data with so-called tags. It can be used to specify other languages, their alphabets and grammars.

Bibliography

[1] *DWARF Project Homepage*. http://www.augmentedreality.de.

[2] *Request For Comments*. http://www.rfc.net.

[3] *M. Abrams, C. Phanouriou, A. Batongbacal, S. Williams, and J. Shuster. UIML: An Application Independent XML User Interface Language.* http://www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html.

[4] *ploticus Homepage*. http://ploticus.sourceforge.net.

[5] *Log4J White Paper*. http://jakarta.apache.org/log4j/docs/manual.html.

[6] W. AALST, *The Application of Petri Nets to Workflow Management*, The Journal of Circuits, Systems and Computers, 8 (1998), pp. 21–66.

[7] K. AHLERS, A. KRAMER, D. BREEN, P. CHEVALIER, C. CHRAMPTON, E. ROSE, M. TUCERYAN, R. WHITAKER, and D. GREER, *Distributed Augmented Reality for Collaborative Design Applications*, Eurographics '95 Proceedings, Maastricht, (1995).

[8] R. T. AZUMA, *A Survey of Augmented Reality*, Presence, 6 (1997), pp. 355–385.

[9] M. BAUER, *Distributed Wearable Augmented Reality Framework (DWARF) Design and Implementation of a Module for the Dynamic Combination of Different Position Tracker*, Master's thesis, Technische Universität München, 2001.

[10] M. BAUER, B. BRÜGGE, G. KLINKER, A. MACWILLIAMS, T. REICHER, S. RISS, C. SANDOR, and M. WAGNER, *Design of a Component-Based Augmented Reality Framework*, In IEEE and ACM International Symposium on Augmented Reality, (2001).

[11] B. BELL and S. FEINER, *Dynamic Space Management for User Interfaces*, in UIST, 2000, pp. 239–248.

[12] E. BERGMAN, *Information Appliances and Beyond*, Morgan Kaufmann Publishers, 2000.

[13] G. BLASKO and S. FEINER, *A Menu Interface for Wearable Computing*, 6th International Symposium on Wearable Computers (ISWC 2002), (2002), pp. 164–165.

[14] D. A. BOWMAN and C. A. WINGRAVE, *Design and Evaluation of Menu Systems for Immersive Virtual Environments*, in VR, 2001, pp. 149–156.

[15] B. BRÜGGE and A. H. DUTOIT, *Object-Oriented Software Engineering. Conquering Complex and Changing Systems*, Prentice Hall, Upper Saddle River, NJ, 2000.

[16] R. CAREY and G. BELL, *The Annotated Vrml 2.0 Reference Manual*, Addison-Wesley Pub Co, 1997.

[17] J. M. CARROLL, *Human-Computer Interaction in the New Millennium*, Addison-Wesley Pub Co, 2001.

[18] M. K. DALHEIMER, *Programming with Qt*, O'Reilley Verlag GmbH & Co. KG, 2002.

[19] F. ECHTLER, H. NAJAFI, and G. KLINKER, *FixIt*, Demonstration at the International Symposium on Augmented and Mixed Reality (ISMAR 2002), Darmstadt, Germany, 2002.

[20] S. FEINER, B. MACINTYRE, and T. HÖLLERER, *Wearing It Out: First Steps Toward Mobile Augmented Reality Systems*, First International Symposium on Mixed Reality (ISMR 1999), (1999).

[21] M. FIORENTINO, R. DE AMICIS, G. MONNO, and A. STORK, *Spacedesign: A Mixed Reality Workspace for Aesthetic Industrial Design*, In Proceedings of the IEEE and ACM: ISMAR 2002, (2002).

[22] G. W. FITZMAURICE, H. ISHII, and W. BUXTON, *Bricks: Laying the Foundations for Graspable User Interfaces*, in CHI, 1995, pp. 442–449.

[23] M. FJELD, M. BICHSEL, and M. RAUTERBERG, *BUILD-IT: A Brick-Based Tool for Direct Interaction*, in D. Harris (ed.) Engineering, Psychology and Cognitive Ergonomics, vol. 4, 1986, pp. 205–212.

[24] M. FJELD, N. IRONMONGER, S. G. SCHÄR, and H. KRUEGER, *Design and Evaluation of Four AR Navigation Tools Using Scene and Viewpoint Handling*, in Proceedings of INTERACT 2001, Eighth IFIP TC.13 Conference on Human-Computer Interaction, Tokyo (JP), 2001.

[25] M. FJELD, S. G. SCHÄR, D. SIGNORELLO, and H. KRUEGER, *Alternative Tools for Tangible Interaction: A Usability Evaluation*, in Proceedings of IEEE and ACM International Symposium of Mixed and Augmented Reality (ISMAR 2002), 2002, pp. 157–166.

[26] T. GILB, *The "Impact Analysis Table" applied to Human Factors Design*, in Proceedings of Interact '84, First IFIP Conference on Human-Computer Interaction (London, September 4-7, 1984), vol. 2, 1984, pp. 97–101.

[27] S. GRIBBLE, M. WELSH, J. VON BEHREN, E. BREWER, D. CULLER, N. BORISOV, S. CZERWINSKI, R. GUMMANDI, J. HILL, A. JOSEPH, R. KATZ, Z. MAO, S. ROSS, and B. ZHAO, *The Ninja Architecture for Robust Internet-Scale Systems and Services*, Computer Networks 35, (2001).

[28] S. HENNAUER, *Empirical Estimation of Tracking Ranges and Application thereof for smooth Transition between two Tracking Devices*. Systementwicklungsprojekt, Technische Universität München, 2003.

[29] C. HESS, M. ROMAN, and R. CAMPBELL, *Building Applications for Ubiquitous Computing Environments*, Pervasive 2002, (2002).

[30] O. HILLIGES, *Development of a 3D-View Component for DWARF based Applications*. Systementwicklungsprojekt, Technische Universität München, 2003.

[31] H. KATO, M. BILLINGHURST, and I. POUPYREV, *ARToolKit version 2.33 Manual*, 2000. Available for download at http://www.hitl.washington.edu/research/shared_space/download/.

[32] B. W. KERNIGHAN and R. PIKE, *The Practice of Programming*, Addison Wesley, 1999.

[33] K. KIYOKAWA, M. BILLINGHURST, S. HAYES, A. GUPTA, Y. SANNOHE, and H. KATO, *Communication Behaviors of Co-located Users in Collaborative AR Interfaces*, in Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2002), 30 Sept. - 1 Oct., 2002, Darmstadt, Germany, IEEE Press, Los Alamitos, CA, 2002, pp. 139–148.

[34] G. KLINKER, A. H. DUTOIT, M. BAUER, J. BAYER, V. NOVAK, and D. MATZKE, *Fata Morgana – A Presentation System for Product Design*, in International Symposium on Aumgented and Mixed Reality ISMAR 2002, 2002.

[35] M. KURZAK, *Architecture and Urban Planning*, Master's thesis, Universität Stuttgart, 2003.

[36] R. LANGENDIJK, *The TU-Delft research program "Ubiquitous Communications"*, 21st Symposium on Information Theory, (2000).

[37] F. LOEW, *Context-Aware Service Selection realized by the AR Toolkit*. Systementwicklungsprojekt, Technische Universität München, 2003.

[38] K. LYONS and T. STARNER, *Mobile Capture for Wearable Computer Usability Testing*, in Proceedings of IEEE International Symposium on Wearable Computing (ISWC), October 08-09 2001, Zurich, Switzerland, 2001, pp. 69–76.

[39] A. MACWILLIAMS, *Using Ad-Hoc Services for Mobile Augmented Reality Systems*, Master's thesis, Technische Universität München, 2001.

[40] A. MACWILLIAMS, T. REICHER, and B. BRÜGGE, *Decentralized Coordination of Distributed Interdependent Services*, in ACM/IFIP/USENIX International Middleware Conference, IEEE DS Online, June 2003, Technische Universität München, 2003.

[41] A. MACWILLIAMS, C. SANDOR, M. WAGNER, M. BAUER, G. KLINKER, and B. BRÜGGE, *Herding Sheep: Live System Development for Distributed Augmented Reality*, In Proceedings of the IEEE and ACM: ISMAR 2003, (2002).

[42] D. J. MAYHEW, *The Usability Engineering Lifecycle*, Morgan Kaufmann Publishers, 1991.

[43] F. MICHAHELLES, *Designing an Architecture for Context-Aware Service Selection and Execution*, Master's thesis, Ludwig–Maximilians–Universität München, January 2001.

[44] P. MILGRAM, KISHINO, and F. A, *Taxonomy of Mixed Reality Visual Displays*, IECE Trans. on Information and Systems (Special Issue on Networked Reality), E77-D (1994), pp. 1321–1329.

[45] A. OLWAL, *Unit - A Modular Framework for Interaction Technique Design, Development and Implementation*, Master's thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, 2002.

[46] B. PIPER, C. RATTI, and H. ISHII, *Illuminating Clay: A 3-D Tangible Interface for Landscape Analysis*, in Proceedings of CHI 2002 on Human Factors in Computing Systems, ACM Press, 2002.

[47] D. PUSTKA, *Visualizing Distributed Systems of Dynamically Cooperating Services*. Systementwicklungsprojekt, Technische Universität München, 2003.

[48] T. REICHER, *Framework for Adaptable Augmented Reality Systems*, PhD thesis, Technische Universität München, 2003. To be Published.

[49] T. REICHER and T. KOSCH, *Software Design Issues for Experimentation in Ubiquitous Computing*, in The Second Workshop on Artificial Intelligence in Mobile Systems (AIMS 2001), Seattle, Washington, USA, August 4, 2001, 2001.

[50] G. REITHMAYR and D. SCHMALSTIEG, *Open Tracker - An Open Software Architecture for Reconfigurable Tracking based on XML*, Technical Report, (2000).

[51] G. REITHMAYR and D. SCHMALSTIEG, *Mobile Collaborative Augmented Reality*, In Proceedings of the IEEE and ACM: ISAR 2001, (2001).

[52] J. REKIMOTO and Y. AYATSUKA, *CyberCode: Designing Augmented Reality Environments with Visual Tags*, in Proceedings of DARE 2000, 2000.

[53] J. REKIMOTO and M. GREEN, *The InformAtion Cube: Using Transparency in 3D Information Visualization*, in Proceedings of the Third Annual Workshop on Information Technologies & Systems (WITS'93), 1993, pp. 125–132.

[54] S. RISS, *A XML based Task Flow Description Language for Augmented Reality Applications*, Master's thesis, Technische Universität München, 2000.

[55] M. ROMAN, C. HESS, R. CERQUEIRE, A. RANGANTHAN, R. CAMPBELL, and K. NAHRSTEDT, *A Middleware Infrastructure to Enable Active Spaces*, IEEE Pervasive Computing, (2002).

[56] J. RUBIN, *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, John Wiley & Sons, 1994.

[57] J. RUMBAUGH, I. JACOBSON, and G. BOOCH, *The Unified Modeling Language Reference Manual*, Addison Wesley, Reading, MA, 1999.

[58] C. SANDOR, *CUIML: A Language for the Generation of Multimodal Human-Computer Interfaces*, Master's thesis, Technische Universität München, 2000.

[59] D. SCHMALSTIEG, A. FUHRMANN, G.HESINA, Z. SZALAVARI, L. M. ENCARNACAO, M. GERVAUTZ, and W. PURGATHOFER, *The Studierstube Augmented Reality Project*, Technical Report, (2000).

[60] B. SHNEIDERMAN, *Designing the User Interface*, Addison-Wesley Publishing, 1997.

[61] A. SMAILAGIC, D. P. SIEWIOREK, R. MARTIN, and J. STIVORIC, *Very Rapid Prototyping of Wearable Computers: A Case Study of Custom versus Off-the-Shelf Design Methodologies*, in Design Automation Conference, 1997, pp. 315–320.

[62] F. STRASSER, *Personalized Ubiquitous Computing with Handhelds in an Ad-Hoc Service Environment*. Systementwicklungsprojekt, Technische Universität München, 2003.

[63] B. STROUSTRUP, *The C++ Programming Language*, Addison-Wesley Pub Co, 1991.

[64] D. SVANÆS and W. VERPLANK, *In Search of Metaphors for Tangible User Interfaces*, in Proceedings of DARE 2000 on Designing Augmented Reality Environments, ACM Press, 2000, pp. 121–129.

[65] A. TANG, C. OWEN, F. BIOCCA, and W. MOU, *Comparative Effectiveness of Augmented Reality in Object Assembly*, in Proceedings of ACM CHI 2003, April 5 April 10, 2003, Ft. Lauderdale, Florida, US, 2003.

[66] M. TÖNNIS, *Data Management for AR Applications*, Master's thesis, Technische Universität München, 2003.

[67] A. TRIPATHI, *Augmented Reality Application for Architecture*, Master's thesis, University of Southern California, 2000.

[68] B. ULLMER and H. ISHII, *Emerging Frameworks for Tangible User Interfaces*, IBMSJ, 39 (2000).

[69] VARIOUS, *Proceedings of the IEEE International Workshop on Augmented Reality - IWAR 1999*, (1999).

[70] VARIOUS, *Proceedings of the IEEE and ACM International Symposium on Augmented Reality - ISAR 2000*, (2000).

[71] VARIOUS, *Proceedings of the IEEE and ACM International Symposium on Augmented Reality - ISAR 2001*, (2001).

[72] VARIOUS, *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality - ISMAR 2002*, (2002).

[73] M. WAGNER, *Design, Prototypical Implementation and Testing of a Real-Time Optical Feature Tracker*, Master's thesis, Technische Universität München, 2000.

[74] A. WEBSTER, S. FEINER, B. MACINTYRE, W. MASSIE, and T. KRUEGER, *Augmented Reality in Architectural Construction*, in Proceedings of the Third ASCE Congress for Computing in Civil Engineering, 1996.

[75] M. WEISER, *The Computer for the 21st Century. Scientific American*, Scientific American, (1991), pp. 94–104.

[76] J. WÖHLER, *Driver Development for TouchGlove Input Device for DWARF based Applications*. Systementwicklungsprojekt, Technische Universität München, 2003.

[77] B. ZAUN, *A Bluetooth Communications Service for DWARF*. Systementwicklungsprojekt, Technische Universität München, 2000.

[78] B. ZAUN, *Calibration of Virtual Cameras for AR*, Master's thesis, Technische Universität München, 2003.