UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

SELF-ADAPTATION OF MULTIMODAL SYSTEMS

Daniel Filipe Ribeiro da Costa

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

2011

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

SELF-ADAPTATION OF MULTIMODAL SYSTEMS

Daniel Filipe Ribeiro da Costa

DISSERTAÇÃO

Projecto orientado pelo Prof. Doutor Carlos Alberto Pacheco dos Anjos Duarte

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

2011

Acknowledgments

Firstly, I would like to thank my family for supporting me during these five years and giving me the opportunity to continue my studies until I wish. Specifically, I want thank my brother that initiated this journey with me. He joined all the work groups we had and still accompanies me, without him I wouldn't be able to be where I stand. Additionally, I thank my mother who helped to overcome so many obstacles that appeared in our life.

Secondly, I would like to thank my advisor Prof. Carlos Duarte for giving me the opportunity to participate in this project and guidance during the realisation of this work. Thank you for your support in difficult times.

Thirdly, I also would like to thank my friends that I made in FCUL and my older friends. Thank you Bruno Neto for being a good friend and excellent colleague, without your good spirits and professionalism we couldn't have those grades. Special thanks to Pedro Feiteira, Joana Neca, Nádia Fernandes, João Ludovico, Tiago Gonçalves and José Coelho. Additionally, I would like to thank Nelson Dias, Ricardo Constantino, Luisa Cabral and Bruno Ribeiro for all the support given and good moments passed.

Lastly but not least, I would like to thank LaSIGE, HCIM and AbsInt for the support and equipment and workstation put available.

To my mother Filomena Costa

Resumo

Este documento centra-se nos Sistemas Multimodais Adaptativos e suas aplicações no melhoramento da acessibilidade das interfaces. Pessoas idosas ou com algum tipo de deficiência, seja associado ou não à idade, são um grupo de alto risco sujeito à exclusão social. Muitas vezes o acesso a oportunidades ou serviços oferecidos pela sociedade é limitado ou mesmo inacessível para indivíduos com estas características. No caso específico da comunicação pessoa-máquina, um exemplo de falta de acessibilidade resulta das modalidades utilizadas para interagir ou apresentar informação. Se for usada apenas informação visual, alguém com dificuldades visuais não conseguirá interagir ou perceber a informação apresentada, logo é excluída.

Como solução para estes problemas, neste documento apresenta-se GUIDE, um projecto europeu que pretende desenvolver uma framework que permite integrar eficientemente características de acessibilidade nas aplicações. Este projecto foca-se nas plataformas e serviços emergentes para TV e pretende transformar as *Set-Top Box's* em sistemas multimodais adaptativos. São apresentados os objectivos do GUIDE, o papel da Faculdade de Ciências neste projecto e as contribuições desta tese.

No segundo capítulo deste documento é apresentada a noção de Sistemas Multimodais Adaptativos, as vantagens associadas a este tipo de sistemas, arquitectura e descrição dos vários componentes. Um sistema multimodal é um sistema que disponibiliza várias modalidades para interagir (voz, gestos, etc), sendo possível a utilização de apenas uma ou várias modalidades em simultâneo. Estes sistemas não só oferecem ao utilizador uma comunicação mais natural como permitem alternativas tanto na maneira de interagir como de apresentar a informação a pessoas que, devido às suas características, de outra maneira não conseguiriam. A outra característica destes sistemas é a adaptação, que pode ser descrita como um método para aumentar a usabilidade de uma aplicação em termos de eficiência, eficácia e facilidade de uso. Este método tráz benefícios para a interacção do utilizador em diversas situações e pode ser aplicado em diferentes níveis de automação apresentados neste documento, sendo o mais interessante a auto-adaptação.

Geralmente, a arquitectura de um sistema deste tipo é composta por reconhecedores para as modalidades de input, sintetizadores para as modalidades de output e entre eles, um conjunto de componentes responsável pela integração multimodal do sistema. Esses componentes são: um módulo de fusão (para os inputs) e outro de cisão (para outputs),

um gestor de diálogo (*Dialogue Manager*) e um gestor de contexto. A descrição e função de cada componente são detalhadas também no capítulo 2.

Esta tese dá especial destaque ao *Dialogue Manager*, cuja responsabilidade é coordenar a actividade dos vários componentes do sistema, manter a representação do estado actual do diálogo, actualizar o contexto do diálogo com base na informação interpretada na comunicação e decidir que conteúdo deve ser apresentado e quando, deixando a forma como é apresentado para o módulo de cisão. Das várias abordagens existentes para a implementação do *Dialogue Manager*, a que vem de encontro aos requisitos do projecto e está descrita neste documento chama-se *"Frame-based approach"*. Esta implementação aplica a analogia do preenchimento de um formulário, ou seja, interpreta o diálogo de um certo estado da aplicação como a necessidade de se verem realizadas certas condições para efectuar uma acção. Cada estado da aplicação pode conter vários formulários e o preenchimento de cada um deles resulta numa acção, que normalmente leva a estados diferentes da aplicação. Contudo, também é possivel haver acções que são independentes da aplicação, como por exemplo pedir ao GUIDE para aumentar o volume. Estes formulários são instanciados automaticamente pelo Dialogue Manager. Pelo contrário, os formulários relativos à aplicação têm de ser inferidos a partir de uma representação da interface da aplicação.

Para realizar um projecto deste tipo é preciso conhecer melhor os futuros utilizadores. Por isso, foram realizados dois estudos feitos com utilizadores-alvo que pretendiam entender como os idosos reagiriam a novas maneiras de interagir com a sua televisão bem como as suas preferências em relacção à apresentação da informação e dos elementos interactivos. As descobertas feitas são relatadas nesta tese e serviram não só para ajudar no desenvolvimento do *Dialogue Manager* mas também todos os outros componentes envolvidos. Outro dos principais objectivos destes estudos foi perceber que diferentes agrupamentos se pode criar para caracterizar e agrupar futuros utilizadores. Para esse fim foi implementado um protótipo que integra diferentes dispositivos para interacção (ex: *WiiMote*, *Kinect*) e que permite a criação de interfaces multimodais recorrendo à descrição dos elementos que desejamos ver no ecrã num ficheiro XML.

Os sistemas adaptativos dependem de informação guardada em modelos como base para aplicar a adaptação. Existem diferentes modelos tendo em conta os diferentes tipos de informação que se quer manter: modelo de utilizador, modelo da tarefa, modelo de domínio, modelo de diálogo, modelo de ambiente, etc. Desses modelos destaca-se o modelo de utilizador pois é o mais importante no campo dos sistemas adaptativos. O modelo de utilizador é a representação do conhecimento e preferências que o sistema "acredita" ser o que o utilizador possui. Toda esta informação é importante para melhorar a interacção e portanto deve ser mantida e actualizada. Neste documento são também descritas as técnicas para a aquisição desse conhecimento, existindo dois processos diferentes para o fazer: explicitamente e implicitamente. Observando o utilizador e o seu compor-

tamento é possível obter as informações implicitamente. Perguntando directamente ou permitindo ao utilizador alterar o modelo de utilizador a aquisição de informação está a ser feita explicitamente.

Neste documento é apresentado um protótipo que pretende servir como um tutorial de modo a ensinar ao utilizador como interagir com o sistema mas também obter informações sobre ele de maneira a poder atribuí-lo a um dos agrupamentos encontrados nos estudos feitos anteriormente. Este protótipo obtém essas informações pedindo ao utilizador para efectuar certas tarefas e respondendo a algumas perguntas.

Como foi referido anteriormente, o *Dialogue Manager* bem como outros componentes lidam com as aplicações através de uma representação da interface de utilizador. Neste documento é feito uma comparação entre algumas linguagens que têm este objectivo e é escolhido então o formato que vai ser usado neste projecto.

Para extrair automaticamente essa representação, no capítulo 6 é apresentado uma ferramenta que o faz a partir de aplicações Web, ou seja implementadas em HTML e Javascript. Esta ferramenta analisa a estrutura dos elementos apresentados através da análise do código HTML bem como de algumas propriedades WAI-ARIA, desenhadas com o propósito da acessibilidade. De seguida, percorre e analisa não só o código HTML com também o CSS, de modo a descrever as propriedades visuais dos elementos que vão ser apresentados no écrã.

Finalmente, todo o trabalho resultante desta tese continuará a ser melhorado e avaliado até ao final da duração deste projecto que contará ainda com 1 ano e 3 meses. Durante esse período será realizada a integração dos vários componentes e será feita a migração das versões em PC para *Set-top-box*. No fim é esperado que todos os objectivos a que este projecto se propôs sejam cumpridos e que pelo menos facilite a vida daqueles que são afectados pela falta de acessibilidade nas tecnologias existentes.

Palavras-chave: Adaptação, Acessibilidade, Sistemas Multimodais Adaptativos, GUIDE

Abstract

This thesis focuses on Adaptive Multimodal Systems and their applications on improving user interface accessibility. Disabled and/or elderly people are a group at high risk of social exclusion. The access to the opportunities offered by society is obviously limited if these cannot be reached by persons with impairments or restricted mobility. A more subtle way of exclusion results from the sensory modalities in which they are presented. Therefore, if the presentation of information has only one modality it will exclude people with impairments in that particular sensory modality. As a solution to these problems, this document presents GUIDE, an European funded project which intends to develop a software framework which allows developers to efficiently integrate accessibility features into their applications.

To perform adaptation the system must know the users, their characteristics and preferences. Thus, a prototype was implemented to assist in user trials. These trials had the goal to understand users' interaction patterns as well as to group users with common characteristics.

In order to match a user with its cluster, it was implemented a prototype named User Initialisation Application (UIA) that besides of tutoring the user on how to interact with the system, it asks the user to perform some tasks and answer some questions. When finished the UIA is able to decide which group the user identifies with.

This thesis takes special focus on Dialogue Manager as it is the core component of the system architecture. It coordinates the activity of several subcomponents in a dialogue system and its main goal is to maintain a representation of the current state of the ongoing dialogue. This document presents the design of the Dialogue Manager that will run in the GUIDE framework. Additionally, in order to the Dialogue Manager and the other components understand the applications' user interface it was implemented a tool that extracts a User Interface Markup Language (UIML) from a Web based application.

Keywords: Adaptation, Adaptive Multimodal System, GUIDE, Accessibility

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This introduction summarizes the main issues this document will discuss. It starts with an overview about multimodal interaction and auto adaptive systems. Then continues with the description of the context in which this work is executed. Finally it presents the objectives to achieve and the planning.

1.1 Overview

Multimodal interaction has shown a great development in the past years concerning the multiple ways researchers explored to enhance human-computer communication. Being a more ”human” way of interacting with computers, using speech, gestures and other ”natural” modalities, multimodal interaction is preferred over unimodal interfaces by users [21]. Two main objectives can be set regarding this type of interaction. On one hand, to support and accommodate users' perceptual and communicative capabilities; on the other hand, to integrate computational intelligence in the real world, by offering more natural ways of interaction to humans [13].

With this type of interaction, the choices offered to users increase and the complexity of the applications grows. Also, with the increase of computer users we have to deal with different levels of expertise and characteristics [9]. In order for an application to adapt to each user's requirements we have to consider different aspects.

For instance, users' requirements change with time. Those changes can occur in different time intervals. The requirements can fluctuate during the execution of a task, which means short time periods. However, those variations can also occur within longer periods, as is the case when the user is learning a new skill. Either way, the application has to be able to adapt to those changes. Self-adaptation is a characteristic of adaptive applications, this feature means that the system has the ability to adapt itself to the user's characteristics and needs. Self-adaptation is fundamentally based on the user's profile updates so it can reflect the most current knowledge. Those updates lead to a better and most appropriate reactions from the application to real-time changes in user needs.

One of the most relevant aspects of the auto-adaptation process is the way the knowledge concerning the users is acquired. In this work, implicit and explicit acquisition strategies will be explored. The implicit strategies are preferred, as they are less intrusive. However, it is important to know the user from the start, so we have to find a strategy to explicitly gather information regarding user's characteristics and preferences in order to form a initial model of the user. Additionally, explicit strategies will be reserved for situations where the lack of knowledge must be immediately surmounted , and occasionally, with confirmatory purposes, as part of an intelligent strategy which provides a maximization of the precision of the assumptions made about the user and a minimization of the application's intrusive behaviour.

This work is being developed in the context of a European project named Gentle user interfaces for elderly people, also known as GUIDE. This project intends to be directed to the elderly society and is described with more detail in the following sections.

1.2 Context

This section describes the context of this work. More specifically, the project which it is related to and the focus on the target users' characteristics.

1.2.1 Elderly social exclusion and impairments

Disabled and/or elderly people are a group at high risk of social exclusion due to the physical, legal, financial, and attitudinal barriers from society that they face in their everyday life. The access to the opportunities offered by society is obviously limited if these cannot be reached by persons with impairments or restricted mobility. A more subtle way of exclusion results from the sensory modalities in which they are presented. Therefore, if the presentation of information has only one modality it will exclude people with impairments in that particular sensory modality. Not being able to use a device or service because its input and output channels support only one modality is a serious restriction of one's everyday life [27]. Ageing and accessibility are two subjects that are highly correlated. Providing accessibility means removing barriers that prevent people with disabilities from participating in substantial life activities, including the use of services, products, and information. Software that is not accessible to a particular user is not usable by that person.

As Oviatt [22] says, multimodal interfaces "have the potential to expand computing to more challenging applications, to be used by a broader spectrum of everyday people, and to accommodate more adverse usage conditions than in the past". Adaptive multimodal interfaces, giving the user an optional representation of the same information in more than one sensory mode, and besides adapting automatically to the user's requirements, can compensate to a certain degree for cognitive and sensory impairments. Besides alternative

outputs, we have to think about alternative interaction inputs so that no impairment can actually prevent users from using the system.

1.2.2 GUIDE - Gentle User Interfaces for Elderly people

As stated before ageing and accessibility are two subjects that are highly correlated in several contexts, like interacting with electronic devices such as computers, nomadic devices or set-top boxes.

Approximately half of the people over 55 suffer from some type of typically mild visual, auditory, motor or cognitive impairments. For them interaction especially with PCs and other complex ICT devices is sometimes challenging, whereas accessible ICT applications could make much of a difference for their living quality while enabling or simplifying participation and inclusion in their surrounding private and professional communities. Ongoing trends in human computer interaction technology development, such as visual sensors, multi- touch, speech or gyroscopic remote controls have the potential to address these needs. When adapted in a right way, those interfaces could help older people, especially with mild impairments, to interact in an intuitive and supportive manner. However, in order to achieve this goal, several gaps need to be filled from a technological as well as from a dissemination point of view:

- The availability of accessible interfaces capable to adapt to the specific needs and requirements of users with individual impairments is very limited, whereas standard UI designs are not sufficient for the needs of those users.

- Existing accessibility APIs do not provide adaptive multimodal interfaces capable to intelligently address the individual requirements of users with different kinds of impairments. Moreover, the adaptation of accessible user interfaces today is only possible through manual configuration.

- The development and integration of state-of-the-art features for accessible user interfaces is expensive and therefore risky for UI framework providers as dedicated Research and Development efforts are needed including expensive user tests for the accessible UI development.

- The integration of accessible user interfaces is also expensive and risky for service/application providers and their ICT developers. They require special experience and knowledge; moreover, great efforts are needed to undertake the user tests required in accessible UI development.

- Most of the service/application providers do not include accessibility features into their products because of time, knowledge and budget constraints, locking out a large portion of their potential users.

- An open framework for accessible UI development could help overcome the expertise, time and budget gap for UI framework providers. Such a framework would allow them to offer state-of-the-art accessibility features without the need for vast investments in Research and Development as well as service/application providers to include accessibility features on several platforms in a common way.

- The availability of accessible user interfaces being capable to adapt to the specific needs and requirements of users with individual impairments is very limited. Although there are a number of APIs available for various platforms that allow developers to provide accessibility features within their applications, today none of them provides features for the automatic adaptation of multimodal interfaces being capable to automatically fit the individual requirements of users with different kinds of impairments.

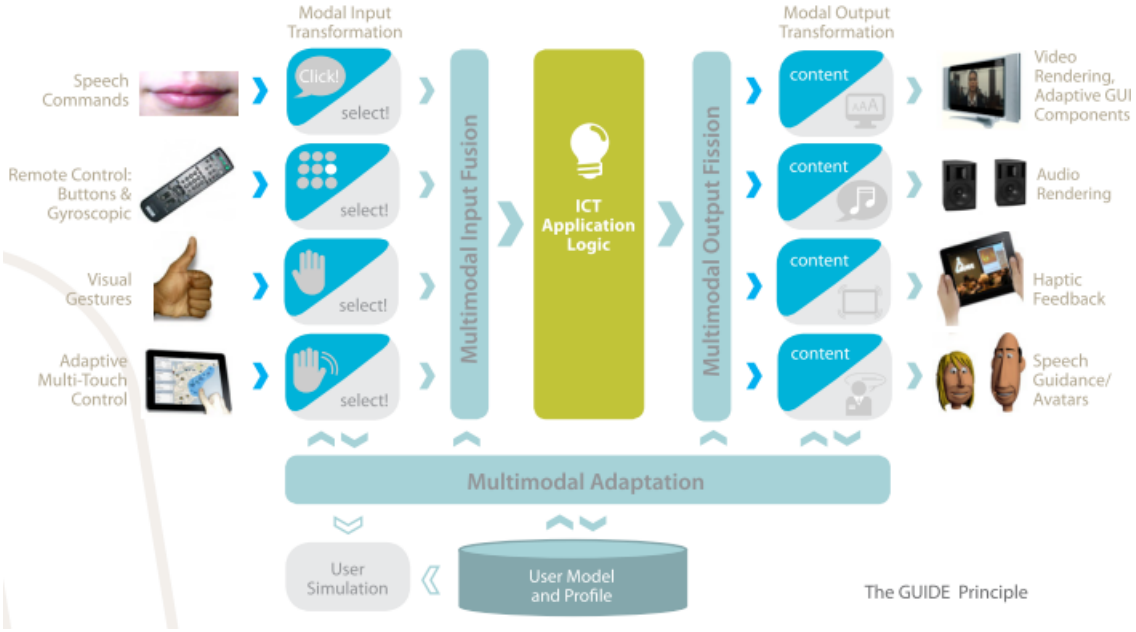

Figure 1.1: Conceptual view of the expected components of GUIDE architecture

As aforementioned, this work will be made in the scope of GUIDE[1]. The European project GUIDE is developing a software framework which allows developers to efficiently integrate accessibility features into their applications. GUIDE puts a dedicated focus on the emerging Hybrid TV platforms and services (Connected TVs, Set-Top Boxes, etc.), including application platforms as Hybrid Broadcast Broadband TV (HbbTV) as well as proprietary middleware solutions of TV manufacturers. These platforms have the potential to become the main media terminals in the users' homes, due to their convenience and wide acceptance. Especially for users of the elderly society, applications such as home

[1]Gentle User Interfaces for Elderly People (GUIDE). `http://www.guide-project.eu`

automation, audio-visual communication or continuing education can help to simplify their daily life, stay connected in their social network and enhance their understanding of the world.

Concerning the contribution of this work to GUIDE, it will be related with the multimodal adaptation module (see figure 1.1), which will be explained further in this document. The following section describes the goals that GUIDE expects to achieve.

Aims

GUIDE aims to promote the use of TV based applications for a specific target population, namely the elderly, but its results should be applicable to the broader population of TV consumers. To reach the elderly population, characterized by a set of age related disabilities, but traditionally also by some reluctance to adopt new technologies, GUIDE instead of relying in new interaction devices, opts for interaction modalities that are already familiar to the target population: speech and gesturing. This multimodal interaction scenario might demand a set of skills from its users, not for the interaction itself, but for the setting up and configuration, that certainly should not be imposed on a regular TV viewer, be him or her elderly or not. As such, in order to achieve the best interaction experience and performance possible from the multimodal interaction set-up, GUIDE has the goal to adapt its features and their operation parameters. Another benefit from presenting users with natural interaction modalities, is that they do not require long learning periods in order to be used, thus overcoming another factor that drives away users from new interaction technologies.

However, the development of adaptive multimodal applications is not common outside research centres, so it certainly cannot be expected that developers of TV based applications have the knowledge required for their development. To address this limitation, GUIDE aims to demand the least possible changes to the current TV based application development methodologies and technologies. Therefore applications will still be developed in the same fashion. GUIDE thus introduces an interaction layer between the application and its user, responsible for presenting the application's contents adapted to its user, but also responsible for translating the natural inputs from the user to an application understandable format. Application developers will still develop their applications considering a TV screen as the output device, and a remote control as the input device and users will be able to consume content in the most adequate modalities for them and with presentations adapted to their capabilities, and also interact through speech and gesturing (pointing or tablet based gestures) besides the remote control.

To be able to generate, in execution time, this adapted multimodal interaction experience, GUIDE needs to know its users. Knowledge about users must be gathered and GUIDE proposes a two-step process. The first step happens the first time a user interacts with the platform and serves two purposes: introducing to the user how the GUIDE sys-

tem will allow them to interact with a TV and collecting, explicitly, enough information about the user to determine what user cluster he or she belongs to. Under the guise of this tutorial purpose, new users will try out the different interaction modalities available to them while GUIDE collects information regarding their abilities. Also, while offering the users the opportunity to personalize their presentation settings (e.g. by asking to select which font size or colours they prefer) GUIDE learns something about their perceptual abilities. The second step is a constant monitoring of the user interaction, allowing to detect requests for presentation changes or difficulties felt in specific interaction scenario, that are immediately used to update the current knowledge about the user (i.e. the user model).

GUIDE ”only” aims to adapt the user interaction with the applications and their interface. This means GUIDE focuses adaptation efforts on two steps of the interaction process (see figure 1.2) input and output.

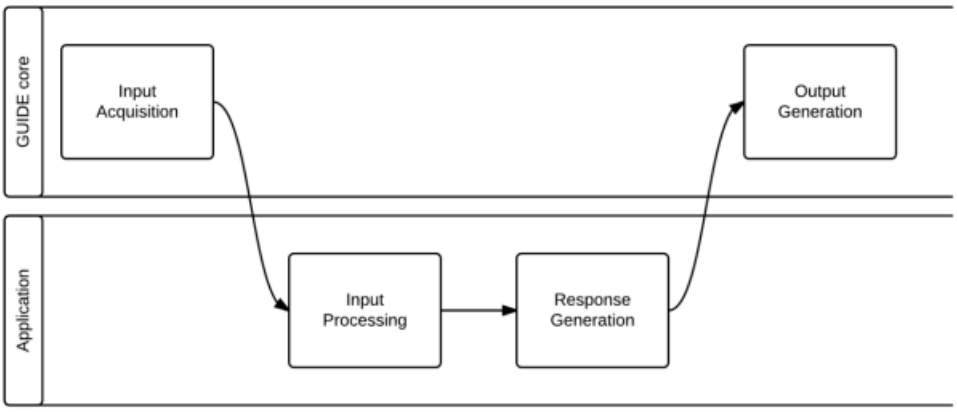

Figure 1.2: GUIDE and application responsibilities in the input and output processing steps

On the input side, GUIDE offers its users additional interaction modalities, like speech, pointing and gesturing, besides the traditional remote control. GUIDE adapts the operation of these modalities to benefit the user (e.g. smoothing cursor movement for users with trembling hands) and the performance of recognition devices (e.g. constantly updating the speech recognizer available commands in any given application state in order to improve recognition accuracy). On the output side, GUIDE renders application content in the most appropriate modality for a given user (e.g. select text or speech synthesis depending on the users visual acuity level) and also adapts parameters in each modality (e.g. increase font size of sound volume, depending on the user’s abilities).

Another goal of this project is to separate between application and interaction, GUIDE makes use of an architecture where applications and the interaction mechanisms reside in different environments. Applications are executed in their environment (e.g. a Web browser for Web applications) while all the interaction is processed inside the ”GUIDE

Core". Communication between the application's environment and the GUIDE Core is achieved through an environment specific interface, developed by whoever is responsible for the application's execution environment. This interface is responsible for implementing an API that allows the application environment to exchange information with the GUIDE Core, but that interface is independent of the application, meaning that any application can be executed in the same environment.

For the GUIDE Core to be able to handle different execution environments, it cannot be constrained by specific application languages (e.g. HTML in the case of Web applications). As a result, one of the responsibilities of the interfaces between applications and GUIDE Core is the translation of the application state (e.g. a Web page) into an execution platform independent language. Therefore, another objective is to find a representation of the user interface that doesn't depend on the application's language.

The multimodal adaptation process described above, even if only targeting the interaction aspects, might be a cause of concern for companies producing applications for TV, which may feel they lack control over presentation aspects of their applications. To mitigate the possible concerns, GUIDE offers three levels of adaptation, representing a "continuum" of intervention on application rendering: augmentation, adjustment and replacement. In the augmentation level, the application keeps the sole responsibility for visual rendering of the UI while GUIDE augments the presentation with other modalities (e.g. present audio content in parallel). In the adjustment level the application outputs its visual rendering of the UI, which is intercepted by GUIDE, who adjusts parameters of the visual output (e.g. alter styles to change text colour and size) before rendering. This in addition to what is performed in the augmentation level. In the replacement level, all rendering is handled exclusively by GUIDE.

1.3 Role of FCUL in GUIDE

Being one of the technical partners of this project, FCUL plays an important role in the development of GUIDE. FCUL leads the design of the adaptive multimodal architecture taking in account the requirements of the project. Thus, FCUL is responsible for some components as well as architectural decisions related to GUIDE's core. Specifically, FCUL is in charge of implementing the Multimodal Fusion module, Dialogue Manager (jointly with partners from Fraunhofer) and the Multimodal Fission module. Additionally, FCUL collaborates in the design, preparation and participation of user trials as well as the implementation of a prototype to be used in those studies. FCUL participates also in the integration of several input devices such as remote control, Microsoft Kinect and Nitendo's WiiMote with the GUIDE project.

1.4 My role in GUIDE

Specifically, the main contribution to this project can be pointed out as assistance in the design, implementation and integration of the Dialogue Manager in GUIDE's core and the development of a tool capable of extracting a user interface representation of the application. To achieve this goal several objectives had to be outlined:

- Conduct an extensive research about the architecture of a multimodal system.
- Learn different methods for applying adaptation techniques.
- Search several ways of implementing a Dialogue Manager and identify one that meets the project requirements.
- Implement a multimodal prototype to be tested with end users.
- Analyse the results obtained in the user trials.
- Define how the Dialogue Manager will represent the dialogue state.
- Define how the Dialogue Manager will work together with the other components.
- Summarize and compare several UI representation formats and choose a standard to use.
- Implement a tool that extracts the UI representation from a Web based application.

1.5 Contributions

This section describes the contributions made during this period:

Carlos Duarte, **Daniel Costa**, David Costa, and Pedro Feiteira. Support for inferring user abilities for multimodal applications. 4 Conferência Nacional Interacção 2010, page 2, 2010. [11]

This paper describes a prototype that supports the characterization of the user's abilities by performing tests in three main areas: sensory, physical and cognitive. This prototype doesn't have any recognition technologies as it uses the Wizard-of-Oz approach to simulate the application reactions to the user's actions, useful in the early stages of development. This paper led to two new contributions to the project: the User Trials Application and the User Initialisation Application.

User Trials Application This contribution consists in an application that is able to present a multimodal user interface from a description made in a XML file. This prototype was used to perform user studies in order to understand their reactions to the use of

multimodal interaction, their preferences, their interaction patterns and to find if it is possible to form clusters taking in account their characteristics.

User Initialisation Application This prototype results from the need of the GUIDE system to know its users. This application instructs firstly the user how to interact with the system using different modalities and then gathers explicitly information about the user by asking them to perform some tasks. These tasks resulted from the user trials and measure specific characteristics of the user in order to match them to a cluster.

Daniel Costa and Carlos Duarte. Self-adapting tv based applications. Proceedings of the 14th International Conference on HumanComputer Interaction HCII, pages 357–364, 2011. [8]

This paper describes the state of the art regarding adaptive multimodal interfaces, Dialogue Manager's different approaches and GUIDE's goals. It also describes the TV based applications that will run in the GUIDE environment and how GUIDE will adapt them. This paper results from the research done in order to learn more about adaptive multimodal systems.

Carlos Duarte, Jose Coelho, Pedro Feiteira, David Costa, and **Daniel Costa**. Eliciting interaction requirements for adaptive multimodal tv based applications. Proceedings of the 14th International Conference on HumanComputer Interaction HCII, pages 42–50, 2011. [10]

This paper describes the analysis made over the results from the first user trials made. It presents interesting findings about the users' preferences regarding interaction modalities (using speech commands, gestures or both), user interface presentation preferences and output modalities (audio, text, images, video and avatar).

User Interface Representation study Another contribution made is a compact research over the available formats to represent abstract or concrete user interfaces. This study was done in order to fill one GUIDE requisite: find a standard UI description independent of the application's language and environment. GUIDE uses this representation to infer possible user commands and to perform adaptation in the user interface presentation (increase buttons' size or complement information with different modalities).

User Interface Representation Extraction Component This prototype extracts a representation of the user interface from a Web based application (HTML and CSS). The application's user interface structure is inferred from the elements presented in the HTML code and style information is gathered from the CSS file.

1.6 Planning

This section describes the tasks that were initially planned and the changes made in that planning as well as some justifications for postponing this thesis delivery.

Firstly, the aim of this work was focused in the study of different techniques to perform adaptation and the integration of them with GUIDE. However, it was soon concluded that those techniques could not be integrated and performed by only one component but by the conjunction of all components of GUIDE. Thus, the goal of this work shifted to a specific component of GUIDE core that unifies the others and manages the dialogue between the user and the system. Additionally, several challenges appeared throughout this period that culminated in new tasks and prototypes.

T1 - Implementation of multimodal applications using an existent framework This task had the goal of introducing to multimodal applications as well as to understand the utility of MMX framework to the GUIDE project. A couple of applications were developed, one of them being an Electronic Programme Guide (EPG). However, the outcome that proved to be useful was the communication system that was used, a publisher/subscriber solution.

T2 - Research of existent literature about Adaptive Multimodal systems The objective of this task was to understand what has been done in this area, guidelines for implementing a multimodal system, adaptation strategies, knowledge acquisition techniques and Dialogue Manager implementation approaches.

T3 - Implementation of a multimodal prototype for user trials This task was not initially predicted, but it was needed to assist in a scheduled user trials for the project. Nevertheless, this task was very useful to learn how to implement a multimodal system from scratch as well as to understand from the results how users accept new modalities to interact with the TV. Additionally it resulted in interesting findings about the dialogue management. From these trials also resulted clusters that represent different kinds of users with different needs.

T4 - Implementation of the User Initialisation Application This task relates with the need of explicitly gather the user's abilities and preferences in order to match with a cluster. The result was an application that serves as a tutorial as well as a means to understand which cluster the user fits.

T5 - Design of the Dialogue Manager This task had the goal to define which of the approaches researched in T2 to implement the Dialogue Manager fits better with GUIDE requirements and how to implement it. Additionally, it also defined the events that this component receives or sends. This task took more time than it was predicted due to several architectural decisions made by the GUIDE's consortium.

T6 - Compare User Interface Representation formats This task resulted from the Dialogue Manager design as DM needs to make a representation of the current dialogue and state of the application. Therefore, this comparison was needed to understand the better format to be applied in GUIDE.

T7 - Implementation of the User Interface Extraction Component (UIREC) This implementation emerged from the necessity to extract from Web based applications the UI representation chosen in T6. Where and when this extraction was supposed to be done also suffered some changes which provoked some schedule extensions.

T8 - Writing of the final report Because of the several delays of some decisions regarding the GUIDE project, this task had to be postponed and therefore be concluded after the 9 month plan.

1.7 Document Structure

This remainder of this document is structured in the following way:

2. Adaptive Multimodal Interfaces This chapter resumes some of the research done in the area of multimodal systems and adaptation techniques. It shows the advantages of using different modalities (independently or simultaneously) to interact with a system. It also describes a generic architecture of an adaptive multimodal system and the functionality of each component. Finally it presents the several approaches to implement a Dialogue Manager depending in the context where the system is inserted and complexity required.

3. User Trials Application This chapter describes a multimodal prototype used in the user trials to collect interesting data about the target users. This chapter also presents the findings related to the diversity of users' characteristics and behaviours and the utility of dialogue management.

4. User Initialization Application This chapter describes the need of an application that tutors the user to the interaction devices available and at the same time evaluates the user's capabilities and preferences. A prototype of such application is presented, implementing some basic tasks based on the information it needs to link the user to a cluster.

5. GUIDE's Dialogue Manager This chapter introduces the GUIDE core architecture and its components as well as the information that flows between them. Then it describes which approach was chosen to implement the Dialogue Manager, how it works and which events does it subscribe or publish.

6. User Interface Representation Extraction Component This chapter describes a tool developed in order to extract a representation of the user interface of any application in an understandable manner for GUIDE's components. It reports the several markup languages taken in account to represent the UI and a prototype of this component.

7. Conclusion This chapter resumes the conclusions of all the work presented in the document.

Chapter 2

Adaptive Multimodal Interfaces

This section describes some basic notions about adaptive multimodal interfaces and their architecture, adaptation methods and knowledge acquisition techniques.

2.1 Multimodalities

Modality can be described as a sense through which a human can receive the output of the computer or a sensor or device through which the computer can receive the input from the human. A system that possesses multiple modalities for communicating either input or output is called a Multimodal system.

Dumas et al. [13] describe multimodal systems as "computer systems endowed with multimodal capabilities of human/machine interaction and able to interpret information from various sensory and communication channels."

According to Oviatt [22], "Multimodal interfaces process two or more combined user input modes (such as speech, pen, touch, manual gesture, gaze, and head and body movements) in a coordinated manner with multimedia system output. They are a new class of interfaces that aim to recognize naturally occurring forms of human language and behavior, and which incorporate one or more recognition-based technologies (e.g. speech, pen, vision)". By using different types of input and output (e.g. Speech, Gestures), Multimodal interaction offers to the user a more natural way to communicate with a machine as this type of interaction has the potential to make the communication similar to what people are used to when interacting with others. Multimodal systems have the capability to enhance human/computer interaction as they offer robustness (due to the combination of different partial information sources) and flexible personalization (based on user and context) [13].

Multimodal interfaces have the potential to give users more expressive power and flexibility, as well as better tools for controlling sophisticated visualization and multimedia output capabilities [22]. Moreover, a multimodal interface offers the user freedom to use a combination of modalities or switch to a better-suited modality, depending on specifics

of the task or environment [23]. Therefore, we can conclude that multimodality has the advantages of synergy (i.e. combination of modalities to achieve a goal), robustness, flexibility and redundancy.

Since Bolt's "Put that there" system [5] in 1980, many examples of systems that explore the advantages of multimodality can be found. For instance, Zhang et al [33] proposed an interface that uses gaze and speech for interaction. On their work they use speech to give commands and an eye tracker that works passively, monitoring the user behavior. Using the eye tracking input information, the system resolves the ambiguity of user's speech and corrects the speech recognizer. On their evaluation experiment, non-native Japanese participants used the application (which has a Japanese recognizer) and were told to select objects on the screen using a simple speech grammar (color, color+shape, size+color+shape). First the system was tested using unimodality and then using multimodality. The results pointed out that using multimodality gives higher success rate than using a single modality, as shown in figures 2.1 and 2.2.

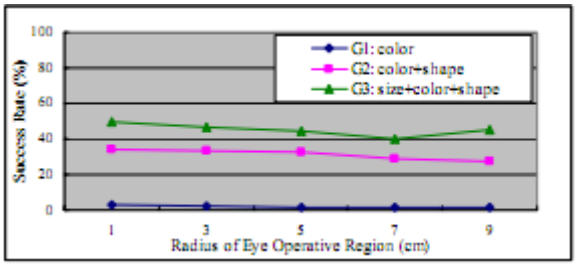

Figure 2.1: Success Rate of speech only modality [33]

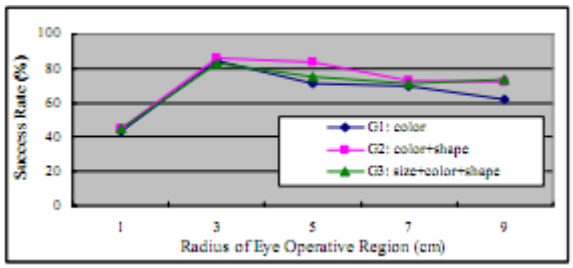

Figure 2.2: Success Rate with multimodality [33]

This section presented the advantages of multimodalities and multimodal user interfaces and their application to human-machine interaction. Despite this, there is one characteristic which is not normally taken in account on these systems but it is becoming, in the recent years, a subject of great matter, which is the capability of the system to adapt to the user's requirements, characteristics and his or her surroundings. In the next section adaptation will be discussed.

2.2 Adaptation

Adaptation is a "method to increase the usability of an application system in terms of effectivity, efficiency, and ease of use" [28]. We can find many possible cases where interface adaptation would benefit the user interaction:

- When the requirements of the user change over time. A user's knowledge and experience evolves over time, from a novice status towards an expert. There are many areas, especially educational systems, where the current proficiency level of the user should be taken into account.
- If a user works in a changing environment. The changes in the environment have to be reflected on the interface. An example is context sensitive help, where the explanations given depend on the current system environment.
- If the system is used by a heterogeneous user population having different requirements. This is the most important reason for having an adaptive system. Users have different needs, preferences, characteristics, knowledge, motivation, goals, etc

There are many applications that include customization components which allow the user to change the preferences or use templates. Although it is the easiest way of providing adaptation mechanisms from the designer's point of view, it is not so easy for the users. This type of customization done by the user can be very difficult and tedious, and sometimes it isn't even done at all, as pointed out by Berry [4]: "Leaving the customization process to the user, may end up reflecting his beliefs, but not his actual practices, since even domain experts have difficulty describing their true knowledge and biases about a domain".

From being totally manual to completely automatic, several levels of automation may be found:

- Self-adaptation. All the steps are done by the interface, without interference from the user.
- User initiated self-adaptation. The user initiates the process of adaptation, but the proposal of alternatives, selection of the best option, and execution are left to the interface.
- User controlled self-adaptation. The interface is responsible for the initiative and proposal of alternatives. The user decides on the best option. The interface executes the adaptation.
- Computer-aided adaptation. The user initiates the adaptation. The interface presents the alternatives. The user selects the best alternative. The adaptation is executed by the interface.

- System-initiated adaptation. The interface initiates the adaptation process. The user is responsible for proposing alternatives, and selecting the best option. The execution can be done either by the user or by the interface.

- Adaptation. The user is responsible for all the steps in the adaptation process, except the execution, which may be done by the interface. GUIDE has the goal to use self-adaptation, making adaptation the system's responsibility as the users don't have enough knowledge to perform it themselves.

2.3 Adaptive Multimodal Systems

Reeves et al [26] points out that "Multimodal interfaces should adapt to the needs and abilities of different users, as well as different contexts of use. Dynamic adaptivity enables the interface to degrade gracefully by leveraging complementary and supplementary modalities according to changes in task and context".

The advantages of multimodal interfaces can be better explored by introducing adaptive capabilities. By monitoring the user's interaction and the context of use, the system can adapt automatically, improving its ability to interact with the user, building a user model based on partial experience with that user. These adaptive capabilities are important when dealing with users with different physical and cognitive characteristics, preferences and knowledge. These systems that use multimodal interactions and have the ability to adapt are called Adaptive Multimodal Systems. An example of such systems is described in [20] where the user's actions and emotions are modeled and then used to adapt the interface and support the user in his or her activity. It integrates a number of natural human communicative modalities including speech, eye gaze direction, face and facial expression and provides decision making and support according to the user's preferences.

GUIDE will be used mainly by elderly people. Therefore putting together the advantages of multimodality and adaptation to overcome these users' requirements and characteristics is an added value.

Next it will be presented what modules an adaptive multimodal interface's architecture should be composed of, and a description of each one's functions and responsibilities.

2.3.1 Architecture

This section describes the modules which compose a generic architecture for an adaptive multimodal interface, following the presented in [13].

In general the architecture of an adaptive multimodal interface is composed by the recognizers for the input modalities, the synthesizers for output modalities and between them there is the "Integration committee" [13]. As shown in figure 2.3, the components

for handling the multimodal integration are: a fusion engine (for inputs), a fission module (for outputs), a dialogue manager and a context manager. The input modalities, such as

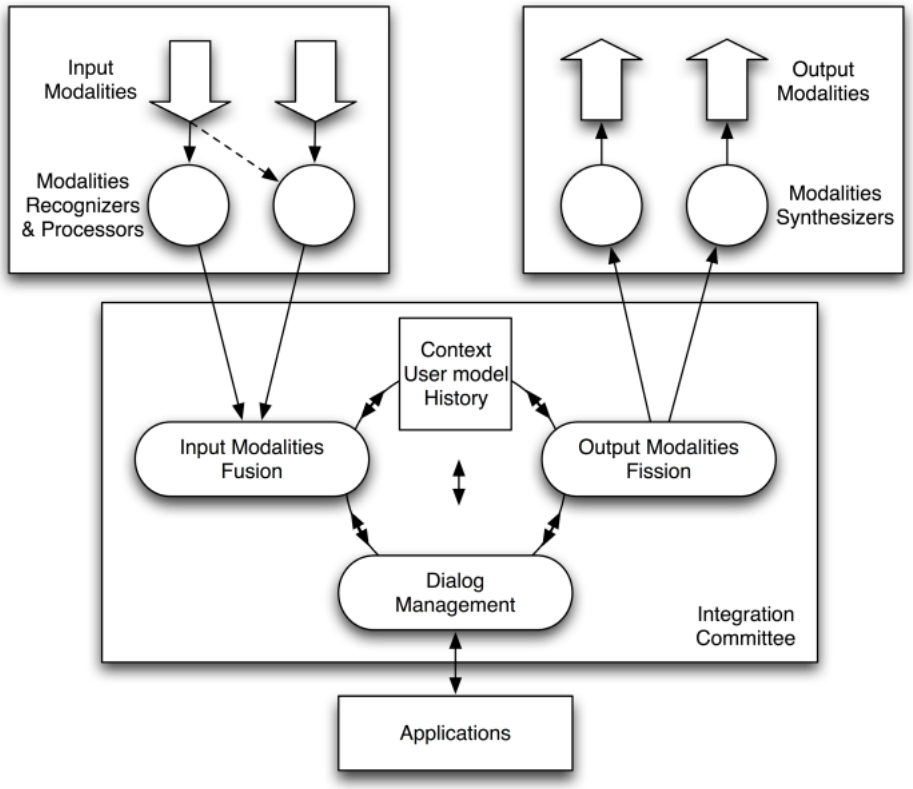

Figure 2.3: General architecture of an Adaptive Multimodal Interface [13]

speech or gestures, are received by the various recognizers. Then the fusion engine is in charge of giving a common interpretation of those inputs. The interpretation given by the fusion engine is passed to the dialogue manager, responsible for identifying the dialogue state, the transition to perform, the action to communicate to a given application, and/or the message to return through the fission component. Then, depending on the user profile and context of use, the fission engine returns the message to the user through the most adequate modality or combination of modalities. Finally, the context manager is responsible for communicating any changes on the environment, context and user profile to the other three components, so they can adapt their interpretations.

Fusion of input modalities

The objective of fusion is to extract meaning from a group of input modalities and pass it to the dialogue manager. In fact, the fusion is one of the features that differentiates multimodal from unimodal interfaces. The fusion of different modalities can be executed at three levels [13]: data level, feature level and decision level.

- Data-level fusion is used when dealing with multiple signals coming from a very

similar source (e.g., two cameras recording from different viewpoints the same scene). With this fusion scheme, no loss of information occurs, as the signal is directly processed. Due to the absence of pre-processing, it is highly susceptible to noise and failure.

- Feature-level fusion, also known as "early fusion", is a common type of fusion when tightly-coupled or time synchronized modalities are to be fused (e.g., speech and lip movements). This fusion is susceptible to low-level information loss but it handles noise better.

- Decision-level fusion (i.e., semantic level or "late-fusion") is the most common type of fusion in multimodal applications. It has the ability to manage loosely-coupled modalities like, for example, pen and speech interaction. Three types of architectures can manage this type of fusion: frame-based architectures, unification-based architectures or hybrid architectures. Failure and noise sensitivity is low with decision-level feature, since the data has been preprocessed. On one hand, this means that decision-level fusion has to rely on the quality of previous processing. On the other hand, unification-based decision-level fusion has the major benefit of improving reliability and accuracy of semantic interpretation, by combining partial semantic information coming from each input mode which can yield "mutual disambiguation"[21].

Fission of output modalities

According to [9] "A multimodal system should be able to flexibly generate various presentations for one and the same information content in order to meet the individual requirements of users and situations, the resource limitations of the computing system, and so forth. Besides the presentation generation, even the presentation content should be selected according to the user's information needs, and the determination of which media to use should be made based on the user's perceptual abilities and preferences".

When we have a set of various output modalities available, such as text-to-speech synthesizers, audio cues, visual cues, haptic feedback or avatars (i.e. animated agents), output selection becomes very difficult to adapt to a context of use (e.g. car, home, work), type of task or type of user (e.g. visually impaired, elderly). Fission techniques [15] allow a multimodal application to generate a given message in an adequate form according to the context and user profiles.

The fission consists of the following three tasks:

- Message construction, where the information to be transmitted to the user is created. The content to be included in the presentation must be selected and arranged into an overall structure.

- Modality selection, specifies the output that is to be realized. Characteristics such as available output modalities, information to be presented, communicative goals of the presenter, user characteristics and task to be performed are forms of knowledge that can be used for output channel selection.
- Output coordination, when multiple output channels are used, layout and temporal coordination are to be taken into account. Moreover, some systems will produce multimodal and cross-modal referring expressions, which will also have to be coordinated.

Dialogue Management

Bui [6] describes a Dialog Manager (DM) as "the program which coordinates the activity of several subcomponents in a dialogue system and its main goal is to maintain a representation of the current state of the ongoing dialogue".

Dialogue Manager is the core module of the adaptive system and therefore the module in which this work will focus. The main tasks of DM are [30]:

- Update the dialogue context on the basis of interpreted communication.
- Provide context-dependent expectations for interpretation of observed signals as communicative behavior.
- Interface with task/domain processing (e.g., database, planner, execution module, other back-end system), to coordinate dialogue and non-dialogue behavior and reasoning.
- Decide what content to express next and when to express it.

A dialogue manager must model to a certain extent two aspects of the interaction between the user and the machine. The first of these is the interaction history which allows user's sentences or actions to be interpreted according to the current context. The second aspect is a model of interaction, controlling the strategy adopted by the system to control the dialogue structure.

Bui [6] considers four different approaches to dialog management:

- Finite-state and frame-based approaches: in this kind of dialog management approach, the dialog structure is represented in the form of a state machine. Frame-based models are an extension of finite-state models, using a slot-filling strategy in which a number of predefined information sources are to be gathered.
- Information state-based and probabilistic approaches: these approaches try to describe human-machine dialog following information states, consisting of five main components: informational components, formal representations of those components, a set of dialog moves, a set of update rules and an update strategy.

- Plan-based approaches: the plan-based approaches are based on the plan-based theories of communicative action and dialog. These theories claim that the speaker's speech act is part of a plan and that it is the listener's job to identify and respond appropriately to this plan.
- Collaborative agents-based approaches: these approaches view dialog as a collaborative process between intelligent agents. The agents work together to obtain a mutual understanding of the dialog. This induces discourse phenomena such as clarifications and confirmations.

These approaches will be discussed further in this document.

Context Management

In order to dynamically adapt the input and output modules to the user's requirements, the system needs to be aware of the many factors that can affect the interaction. The Context Manager has the responsibility to communicate to the other modules the changes in the context of use of the system (e.g., user impairments, preferences, environment, level of expertise) in order to enable the interface to provide complementary and supplementary modalities. The individual differences like age, skill, sensory and cognitive impairments can be captured in a user profile and used to determine the way the interface can adapt to those specific requirements.

2.4 Dialogue Management

This section will start by presenting the models from which the Dialogue Manager (DM) gets the information needed to adapt and the techniques used to obtain that information. Then, it will discuss the many approaches to implement a DM and finally it will discuss the use of Machine Learning on adaptive multimodal interfaces.

2.4.1 Context Models

Adaptive systems rely on information stored in models as basis for adaptation. There are different models concerning different types of information to be stored: user model, task model, domain model, dialog model, environment model, etc. The most important model in the adaptive system field is the user model, as the user is the main driver of adaptation in most systems [9]. Being an integral part of any adaptive system, the user model is a representation of knowledge and preferences which the system "believes" that a user (or group of users) possesses [3]. The user model holds the user's characteristics, behavioral patterns, preferences, etc. All this information is important for the adaptation process and for improving the interaction. Therefore, the user model must have continuous maintenance and be always updated.

A number of general knowledge sources is usually held, updated and shared by the system components [6]:

- Dialogue history: A record of the dialogue so far in terms of the propositions that have been discussed and the entities that have been mentioned. This representation provides a basis for conceptual coherence and for the resolution of anaphora and ellipsis .

- Task model: A representation of the information to be gathered in the dialogue. This record, often referred to as a form, template, or status graph, is used to determine what information has not yet been acquired.

- World model: This model contains general background information that supports any commonsense reasoning required by the system, for example, that Christmas day is December 25.

- Domain model: A model with specific information about the domain in question, for example, flight information.

- User model: This model may contain relatively stable information about the user that may be relevant to the dialogue such as the users age, gender, and preferences (user preferences) as well as information that changes over the course of the dialogue, such as the users goals, beliefs, and intentions (user's mental states).

Robert Kass [18] classifies the potential contents of the user model into these four categories:

- Goals and Plans. A common problem for many applications is determining what the user really wants to achieve by using the system. This is particularly true of natural language systems, since the user frequently does not state his or her goal explicitly. Thus a user modeling system may be required to discern and keep information about the goals of the user. In many cases the user will have a plan for achieving these goals. Some applications, such as intelligent tutoring systems, need to know user plans in order to judge whether they are correct. This is also true for cooperative systems that seek to correct a user's plan if the system recognizes the plan will not achieve the user's goal.

- Attitudes. People have definite opinions about certain things. For a system to interact with the user in a natural manner, it sometimes must be sensitive to the attitudes held by the user. For example, systems that advise the user must be cognizant of user attitudes when making a recommendation, both to avoid suggestions the user will not accept, and to reassure the user that particular requirements are being met.

- Objective properties. Many applications need to know objective properties of the user. Often these properties, such as the user's name or age, can be ascertained without talking to the user. Objective properties could also include the user's physical capabilities, such as whether the user is capable of performing a certain action or test recommended by the system.

- Beliefs. Most applications require some knowledge of the beliefs of the user. These beliefs encompass not only beliefs the user has about the world or about the domain of the application, but also the beliefs the user has about the system and about other agents. Tutoring systems and help systems have a great need for knowledge about the beliefs of the user, since they must be able to accurately judge what the user does and does not know. Cooperative systems in general also need such knowledge to help tailor responses given to the user, to ensure that what the system says is understood by the user and is not misleading.

In the next section it will be pointed out techniques for acquisition of the information to be stored.

2.4.2 Knowledge acquisition techniques

One of the most relevant aspects of the self-adaptation process is the way the knowledge concerning the users is acquired. According to Duarte [9], the process of acquisition of the information needed to build and maintain the user model can be done in an implicit or explicit way. By observing the user's behavior, the interface can obtain information in an implicit way, and use it to enhance the experience and capabilities of the user. This acquisition can be done by observation of direct interaction with the interface, or by the analysis of the information needed by the user. There are two types of information that can be gathered with implicit acquisition: short-term information or long-term information. The short-term information is formed by the current dialogue state (e.g., last sentence typed). This information can be used to give immediate assistance in case of a query to the system. The long-term information consists on the interaction history. For example, the system can assume that every command that has already been used is known to the user or if a set of commands has been used correctly for a number of times, then it can assume the user skill has evolved and increase the level of expertise of that user.

The explicit acquisition of information for the user model can be done in two ways. The application can ask questions directly to the user regarding the information that is lacking, or the user might be allowed to inspect and possibly make changes to the user model. By asking for information rather than deriving it from other sources, this knowledge acquisition technique can be more precise and reach higher levels of accuracy of the information kept in the user model. However, this involves interruptions on the user task and therefore is a more intrusive way of getting the information [9].

An example of the application of these techniques can be seen in [2], where it is described an approach for Personalized Electronic Program Guides to face the information overload and facilitate the selection of the most interesting programs to watch. The system runs on the set-top box and downloads information about the available TV programs from the satellite stream. In order to obtain precise estimates of the individual TV viewer's preferences, their system relies on the management of a hybrid user model that integrates three sources of information: user's explicit preferences; information about viewing preferences of stereotypical TV viewer classes; and user's viewing behavior.

2.4.3 DM Techniques

This section will present several approaches to implement a Dialogue Manager. As should be easily understandable, it is impossible to discuss all approaches here due to the high number of hybrids or extensions.

Finite state-based and frame-based approaches

Finite state models are the simplest models used to develop a dialogue management system. The dialogue structure is represented in the form of state transition networks in which the nodes represent the system's utterances and the transitions between the nodes determine all the possible paths through the network. The Dialogue control is system-driven and all the system's utterances are predetermined. In this approach, both task model and dialogue model are implicit and they are encoded by a dialogue designer. The major advantage of this approach is the simplicity. It is suitable for simple dialogue systems with well-structured task. However, the approach lacks of flexibility, naturalness, and applicability to other domains. [6]

In [25] Pietquin points out the following advantages of this approach:

- Finite-stated method is easier to understand and more intuitive for the designer as a visual, global and ergonomic representation.
- It is much more straightforward to give a visual representation of finite-state as a dialogue is described as a set of states linked by transitions.
- Lots of applications like form-filling, database querying or directory interrogation are more successfully processed this way. [32]

An extension of finite stated-based models, frame-based models, is developed to overcome the lack of flexibility of the finite state models. The frame-based approach, rather than building a dialogue according to a predetermined sequence of system's utterances, takes the analogy of a form-filling (or slot-filling) task in which a predetermined set of information is to be gathered. This approach allows some degree of mixed-initiative and

multiple slot fillings. The task model is represented explicitly and the dialogue model is (implicitly) encoded by a dialogue designer. The frame based approaches have several advantages over the finite state-based approaches: greater flexibility and the dialogue flow is more efficient and natural. However, the system context that contributes to the determination of the system's next action is fairly limited, and more complex transactions cannot be modeled using these approaches. [6]

In [14] a system for monitoring the health status of chronic patients that uses this approach is presented. The users can enter their personal clinical data into a database by periodically calling a telephone service and by engaging in a dialog with it. The dialog takes into account the clinical histories of patients and, if needed, can give advices to them or can alert doctors. The system can handle mixed-initiative interactions through a dialog engine that acquires basic information (e.g. patient personal code, blood pressure, weight, etc.) and each basic piece of information has associated voice prompts and corresponding grammars. The system changes its run-time behavior depending on both the progress of the current call and the clinical history. In this way, an adaptive dialog system is realized.

Information state-based and the probabilistic approaches

In [6], Bui refers that Information state approach and its extensions are an effort to overcome limitations in finite-state based and frame-based approaches. An information state-based theory of dialogue consists of five main components [30]:

- A description of informational components (e.g. participants, common ground, linguistic and intentional structure, obligations and commitments, beliefs, intentions, user models, etc.).
- Formal representations of the above components (e.g., as lists, sets, typed feature structures, records, Discourse Representation Structures, propositions or modal operators within a logic, etc.).
- A set of dialogue moves that will trigger the update of the information state.
- A set of update rules, that govern the updating of the information state.
- An update strategy for deciding which rule(s) to apply at a given point from a set of applicable ones.

The general idea of the information state approach was used for the development of multimodal dialogue systems such as Virtual Music Center [16], MATCH system for multimodal access to city help [17] and Immersive Virtual Worlds [31].

Another extension to the information state approaches is to use probabilistic techniques such as (fully observable) Markov Decision Process (MDP) or a Partially Observable Markov Decision Process (POMDP). The idea is to dynamically allow changing of

the dialogue strategy and the actions of a dialogue system based on optimizing some kinds of rewards or costs given the current state. This is done by modeling the dialogue as a MDP or as a POMDP. Reinforcement learning is then used to learn an optimal strategy. The actions are the system's responses and questions and the rewards are either defined by the designer (high reward for task completion, low punishment for confirmation and questions and so on) or they are provided by the user who is asked to rate the system at the end of each dialogue. [6]

Plan-based approaches

Plan-based approaches support a greater complexity to dialogue modeling than the approaches presented in previous sections. Bui [6] explains that these approaches are based on the view that humans communicate to achieve goals, including changes to the mental state of the listener. The dialogue's input is not only considered as a sequence of words but as performing speech acts [29] and it is used to achieve these goals. Usually, the task of the listener is to discover and appropriately respond to the speaker's underlying plan. The plan-based approaches are based on the plan-based theories of communicative action and dialogue which claim that the speaker's speech act is part of a plan and that it is the listener's job to identify and respond appropriately to this plan.

Plan-based approaches have been criticized on practical and theoretical grounds. For example, the processes of plan-recognition and planning are combinatorically intractable in the worst case, and in some cases they are even undecidable. These approaches also lack of a sound theoretical basis. There is often no specification of what the system should do, for example, in terms of the kinds of dialogue phenomena and properties the framework can handle or what the various constructs like plans, goals, etc. are. [6]

Collaborative agent-based approaches

Collaborative approaches (also known as agent-based dialogue management) are based on viewing dialogues as collaborative process between intelligent agents. Both agents work together to achieve a mutual understanding of the dialogue. Bui [6] explains that unlike the dialogue grammars and plan-based approaches which concentrate on the structure of the task, the collaborative approaches try to capture the motivations behind a dialogue and the mechanisms of dialogue itself.

The advantages of the collaborative approaches are the ability to deal with more complex dialogues that involve collaborative problem solving, negotiation, and so on. But the approaches require much more complex resources and processing than the dialogue grammars and plan-based approaches.

2.4.4 Machine learning

Machine learning techniques play an important role in adaptive multimodal interfaces. Many parts of multimodal systems are likely supported by machine learning. For example, the speech, face and gesture recognizers are domains of interest on machine learning. On the self-adaptive systems field, the machine learning can be useful on the user, task and context modeling.[13]

Machine learning is the use of algorithms that allow computer programs to automatically improve through experience. Because adaptive user interfaces must learn from observing their user's behavior, another distinguishing characteristic is their need for rapid learning. The Dialogue manager exists to oversee and control the entire conversational interaction and execute any number of functions. Ultimately, the dialogue manager prescribes the next action on each turn of an interaction. Because actions taken by the system directly impact users, the dialogue manager is largely responsible for how well the system performs and is perceived to perform by users (i.e. the user experience) [24]. This recommends the use of learning methods that achieve high accuracy from small training sets over those with higher asymptotic accuracy but slower learning rates. On the other hand, the advisory nature of these systems makes this desirable but not essential; an interface that learns slowly will be no less useful than one that does not adapt to the user at all. Still, adaptive interfaces that learn rapidly will be more competitive, in the user's eyes, than ones that learn slowly [19]. An example of a learning method is the reinforcement learning, one the most active research areas in artificial intelligence. Reinforcement learning is a computational approach to learning whereby an agent tries to maximize the total amount of reward it receives when interacting with a complex, uncertain environment. More detailed information about this learning method can be seen in [1]. Other examples of well-known learning methods are Bayesian networks, which use probabilistic models to perform inference and learning; or Decision tree learning, a method that uses a decision tree as a predictive model.

2.5 Discussion

This chapter described the advantages of using multiple modalities to interact. This approach can reduce errors when understanding the user intentions, besides providing different modalities for content presentation can avoid exclusion of certain users with impairments. Therefore, GUIDE must indeed implement multimodality either for input and output as it would help to make the applications more accessible. Another different approach described is the adaptation of the user interface automatically based on the knowledge the system has about the user. In order for self-adaptation to work GUIDE must have a model of the user that contains the information needed to perform adaptation. However that information does not appear magically, so this chapter described different techniques

that GUIDE can use to gather that knowledge implicitly (e.g. using machine learning and observing user's behaviours) or explicitly (e.g. asking directly to the user some information). Another important issue presented is the variety of approaches to implement the central component of the adaptive multimodal system: Dialogue Manager. One must be chosen that meets GUIDE's requirements and limitations (e.g. set-top-box performance, complexity of the algorithms, application's structure).

Chapter 3

User Trials Application

3.1 Introduction

The development strategy for this project had to be clearly supported by a user centred methodology as its target users belong to a very specific demographic segment of the population. User Centred Design focus on users through the planning, design and development phases in order to meet their requirements and needs. This method requires not only to analyse and foresee how users are likely to use a product, but also to evaluate their assumptions with regards to user behaviour in real world tests with actual users. Therefore, a tool had to be built that provides a way to elicit the users interaction requirements and understand how their abilities impact their perception. Also, because GUIDE is a multimodal system this tool has to provide multiple modes of interaction with the view to observe how users integrate and explore those modes when interacting with the system.

The use of low fidelity prototypes made of paper and other non-technological materials were considered but given the wish to acquire reactions to what technology could offer its users as close to reality as possible this approach was discarded. Therefore, we decided to implement a multimodal system, resorting to a Wizard of Oz based approach, that allows to explore individual reactions to interaction means available, and to study usage patterns.

This prototype consists in a series of scripted tests, where each user experiments every device and every modality for selecting items in different menus (both individually and in a combined manner), as well as gives opinion about several different configurations of buttons, interfaces, background colours, font size and colour, and several more interface and interaction aspects (e.g. for menu interaction do you prefer only visual output or both visual and audio output?).

Besides, this implementation gave a perfect practical introduction to multimodal systems and its components. The following sections describe the goals of this prototype and the user trials, architecture and additional technical aspects of the implementation of the aforementioned application and the analysis of the tests results.

3.2 Goals

The User Trials Application is a tool used in the user trials by the end-users of the system so they can learn all possibilities of interaction by experimenting different devices and different ways of interaction with a TV and STB based system, as well as the main tool for gathering relevant information for user clustering so that the system can be prepared to handle different types of users. Therefore the main objective of this application is to assist the partners involved in the user trials.

The goals of the user trials are first of all to understand the users' reactions to new technologies that enable a natural interaction with a TV, observe their behaviour and interaction patterns that can be useful to get new requirements for each component of the GUIDE system and relate their preferences with their impairments and capabilities. Finally, at the end of the user trials the information collected directly by the logs of the application (that saves information related to times of selection and navigation steps in each interface presented to the user) and by the user trial supervisors (user preferences, times of selection, cognitive skills, etc.) is analysed, as these studies aim to find different clusters where the future users will be inserted.

3.3 Architecture

This section describes the architectural aspects of the User Trials Application implementation (see figure 3.1). This system tries to emulate a full multimodal system in the eyes of the user, and it is a good tool to perform a study over any use case as it is fairly flexible in what concerns scenario creation. Users can interact with the system using different modalities individually or simultaneity. Also, information about the interaction actions performed (which targets were selected, how much time, and task completion) are stored in a log file. As the figure shows the log is written by the Graphical User Interface as it holds the informations needed to be stored (e.g. which element was select by the cursor). The main component of this system, developed in Java, is called the Engine and it serves as Dialogue Manager, the central component of a multimodal system (see chapter 2). This component has the goal to setup the states, which are a set of tests that the developer can define in a XML file, and to manage the communication of each component, deciding where each message goes. Which tests to perform, some input simulation and evaluation scores over the users actions are in charge of a Wizard of Oz.

Input and Output modules are integrated using a communication based on subscription and with defined XML messages.

The most interesting feature is the ability to create a multimodal user interface from a XML description. This feature provides an easy way for the experimenters to implement their tests whenever is needed (e.g. performing some tweaks during the trial period) and what they want to see displayed and tested.

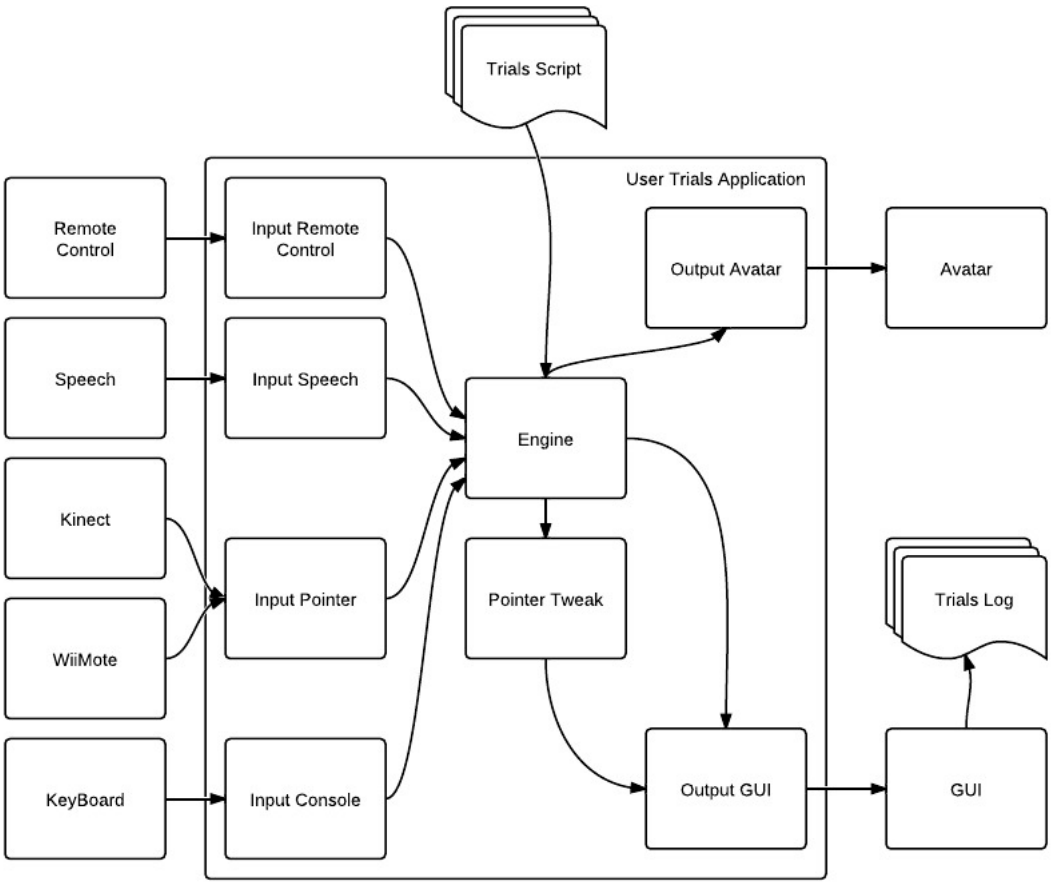

Figure 3.1: Architecture of the User Trials application

3.3.1 XML interface approach

As mentioned above, the developer or any person responsible to create the tests can do it just by building a quite simple XML file. This XML (example in figure 3.2) contains a script of which interactive elements (e.g. buttons, labels, images) it is desired to appear on the screen and their characteristics such as fontsize or background color, what modality should they be rendered, screen information, etc.

```
xml version="1.0" encoding="utf-8" ?>
onfig width="1680" height="1050" filter="warping">
<visual>
  <question filter="" type="visual" number="welcome" screen="a" bgcolor="WHITE">
    <item type="label" synth="false" text="Visual Tests" fontsize="40" bgcolor="WHITE" fgcolor="BLACK" hlcolor="WHITE" x="0.006" y="0.300" width="1200" height="100" />
  </question>
  <question filter="" type="visual" number="Vis1" screen="a" bgcolor="WHITE">
      <item type="label" synth="false" text="Vis1 1/2" fontsize="20" bgcolor="WHITE" fgcolor="BLACK" hlcolor="WHITE" x="0.500" y="0.900" width="1200" height="100" />
      <item type="button" synth="false" text="CIENCIA" fontsize="40" bgcolor="WHITE" fgcolor="BLACK" hlcolor="GOLD" x="0.006" y="0.011" width="250" height="100" />
      <item type="button" synth="false" text="TV" fontsize="40" bgcolor="WHITE" fgcolor="BLACK" hlcolor="GOLD"  x="0.006" y="0.450" width="250" height="100" />
      <item type="button" synth="false" text="DEPORTE" fontsize="40" bgcolor="WHITE" fgcolor="BLACK" hlcolor="GOLD"  x="0.006" y="0.850" width="250" height="100" />
      <item type="button" synth="false" text="MUSICA" fontsize="40" bgcolor="WHITE" fgcolor="BLACK" hlcolor="GOLD"  x="0.800" y="0.011" width="250" height="100" />
      <item type="button" synth="false" text="VIAJE" fontsize="40" bgcolor="WHITE" fgcolor="BLACK" hlcolor="GOLD"  x="0.800" y="0.450" width="250" height="100" />
      <item type="button" synth="false" text="COCINA" fontsize="40" bgcolor="WHITE" fgcolor="BLACK" hlcolor="GOLD"  x="0.800" y="0.850" width="250" height="100" />
  </question>
```

Figure 3.2: XML example describing test screens

This approach provides an easy way to build test screens or even emulate a future application with a multimodal user interface in order to gather users reactions and inter-

action patterns, useful to enhance the software design in early stages of development.

3.3.2 Wizard-of-oz approach

The Wizard-of-Oz approach is a method to simulate the behaviour of a theoretical intelligent and fully functional system. This is done with the help of an experimenter (the ”wizard”) that intercepts all communications between the users and the system and performs the actions that were expected. This method has proven to be very useful to simulate for instance the voice recognizer and let the users feel free to use any words without restrictions. To perform this control over the system, the ”wizard” has a simple console with menu based options. This console offers the ”wizard” the possibility, among others, to load script files, move to next tests, classify the user performance in a task and simulate a target selection.

3.3.3 Input and Output modalities

This system allows the use of several input and output modalities and devices. The majority of them were fully implemented, but the speech recognition for example was simulated by the ”wizard” as previously described. The main output component is called OutputGUI which sends the user interface specification, cursor positions and input actions (e.g. select element) to the Graphical User Interface (GUI). The GUI component is developed in C#, it is responsible to render the graphical user interface (buttons, labels, images, etc), speech and cursor. The other integrated output component is the Avatar, developed by CCG (Centro de Computação Gráfica), one of GUIDE consortium partners, in C++, it is an animated model that can also synthesize speech. For input it was available some gestures and pointer devices, voice commands and a remote control. Some of these devices were added in different versions of the system. Table 3.1 describes the I/O devices available and the actions they perform.

Device	**Input and Output on the User Test Application**
Remote Control	Button selection (input)
Wii Remote + Wii Sensor bar	Pointing, gesture and button selection (input)
Kinnect / Led Camera Sensor	Pointing, gesture and button selection (input)
Avatar Engine	Audio and visual output
Speech Synthesis	Audio output
Simulated Speech Recognition	Audio input (Console input)
Tablet (Apple’s Ipad)	Touch screen input and visual and haptic output

Table 3.1: I/O modalities available

3.3.4 Communication protocols

The communication approach integrated in this system is a publisher/subscriber platform called MessageQueue from Oracle. This approach allows each component to subscribe or publish their messages, and provide a flexible way of adding components simply by publishing their messages in the defined XML format. Those components can be new input or output interaction modules.

3.4 First User Trials

The script included tasks to exercise perceptual, motor and cognitive skills. These tests aimed at exploring how users perceive and prefer visual elements (font size, font and background colors, object placement, etc.) and audio stimulus (volume). Also, a set of tasks to exercise the user's cognitive abilities were included. Furthermore, and even though in some tasks users were free to interact in any way they wished, including combining modalities, we included tasks for users to explicitly use more than one modality, in order to observe what integration patterns may be revealed. The following sections explain these trials setup, tests and the analysis over the findings.

3.4.1 Setup

The first trials were conducted over a period of one week. Nine elderly people participated in these trials: five men and four women. Their average age was 74.6 years old, with the youngest being 66 years old and the oldest being 86 years old. Sessions lasted about 40 minutes, in which participants familiarized themselves with the available means of interaction and executed the required tasks.

The system was installed in a PC with two screens: one regular monitor for the "wizard" and one TV for the user. The devices available for interaction in these first trials were the WiiMote and a Led Camera Sensor, which detects user's hand motion. Also, speech commands were accept as input.

3.4.2 Tests implemented

As mentioned previously the trial's script took in consideration visual, audio and motor requirements. Moreover, in some tasks participants were free to elect how they wished to interact with the User Trials Application while in other tasks they were specifically requested to interact with a specific modality or combination of modalities. In most of the script's questions participants had to perform the same task with different modalities or with a different rendering parameter and after performing the task they had to report their favourite configuration for that task. In some other questions participants were just asked to select between one of differently rendered presentations without having to explicitly

perform a task. Figure 3.3 shows a user performing one of the tasks, using the motion recognizer to point at the screen.

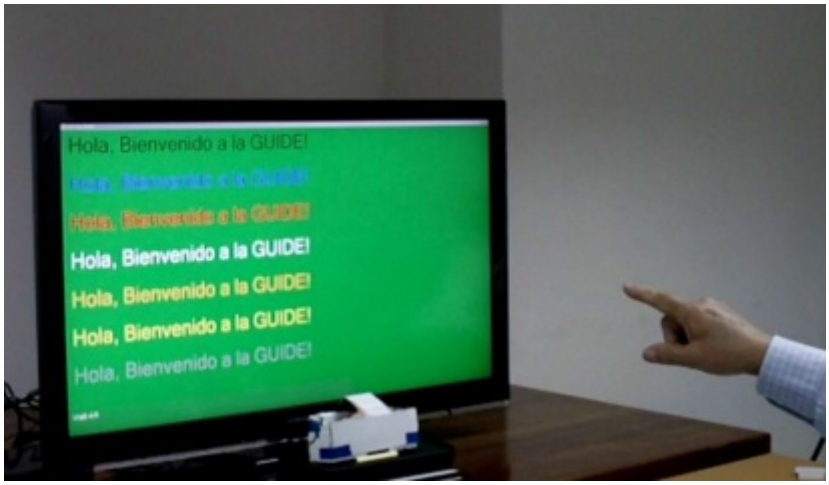

Figure 3.3: User performing a task

The trials begun with a simple explanation task, where participants were asked to select one of four options presented on screen as four buttons. They had to do it through finger pointing, using Wii remote to point, speech and with a combination of speech and one of the pointing options. Afterwards, different presentations were explored: target placement (near the center or near edges of the screen), target size, target spacing, target color, text size and text color. Only for the color presentations participants simply had to select their preferred option (see example in figure 3.3). In all other, they had to perform a task, such as selecting one of the targets or reading aloud the text. The trial then proceeded with an audio rendering task where participants were asked to repeat the text that was rendered to them through a TTS (text to speech) with different volume levels. These were followed by motor tests, where users had to perform gestures (not simply pointing) both without surface support or using a tablet emulating surface. Afterwards, users had to perform a selection task in all combinations of input (options presented visually, aurally or both combined) modalities. Finally, they had to assess what would be their preferred modality to be alerted to an event when watching TV and when browsing photos on the TV. Options included on screen text, TTS, the avatar or combinations of the three. the test ended with a comparison between the avatar and a video of a person presenting the same content, to assess the avatar's ability to generate empathy with future system users.

Table 3.2 summarizes the type of tests, the tasks and the modalities users could use.

3.5 Second User Trials

The User Trials application suffered some improvements taking in account the observations made by the GUIDE experimenters during the trial and the conclusions made from

Type of tests	Tasks	Devices/Modalities
Modalities and devices experimentation	Answering to questions related with preferences of interaction, experimentation of each device and modality. Menu items selection and navigation.	Input: WiiMote, Motion sensor. Output: visual menus and Avatar.
Visual capabilities and preferences	Answering to questions related with interface visual configuration (font size and color, etc). Menu item selection and navigation.	Input: one or more devices choosen by the user. Output: visual menus.
Audio capabilties and preferences	Answering to questions related with audio preferences (audio volume).	Input: Speech. Output: Audio.
Cognitive capabilities	Localization of different items on the screen (cognitive scientific tests), measuring time of response.	Input: Speech, WiiMote, Motion Sensor. Output: Visual menus and pictures
Motor capabilities and preferences	Performing gestures. Menu item selection and navigation. Interacting with the tablet. Answering to questions related with motor preferences and pointing mechanisms.	Input: WiiMote, Motion Sensor and tablet. Output: Visual menus.
Avatar preferences	Interacting with avatar. Answering to questions related with avatar preferences.	Input: One or more devices chosen by the user. Output:Visual menus and Avatar.
Multimodal preferences	Menu item selection and navigation. Simulation of application contexts of use. Answering to questions related with multimodal interaction and references.	Input:One or more devices chosen by the user. Output: Visual and audio menus and avatar.

Table 3.2: Tests, tasks and modalities

the first trial.

One of the enhancements implemented was the inclusion of adaptation algorithms for cursor movement. Those algorithms perform different strategies to ease the cursor movement by the users. The first algorithm is called ”Exponential Averaging” and what it does is averaging the current cursor position values (x and y) with the previous ones, giving users the feel that cursor movement is smoother. Another algorithm is named ”Damping” and it intends to oppose the fast movement of the cursor, so if the user performs a sudden movement it will suffer some attrition reducing the speed. Finally, the last algorithm is ”Warping” and it uses the information about the elements displayed on the screen to ”attract” the cursor to them. For instance, if the user moves the mouse near a button, it will slowly bent towards the button’s center.

Another improvement made was the enhancement of the ”wizard”’s console, as the experimenter wasn't always a technical expert the console needed to be more simple and fool-proof. Additionally, it was added more options such as saving the results on the log whenever the ”wizard” wanted and not necessarily in the end of the trial and the possibility to select tests by name.

Related with tests, experimenters reported the need to randomize tests screens in order to not influence the participants' answers. Also, some bugs were corrected and new tests made.

New input devices were added in these trials: Microsoft Kinect and a Remote Control. And a new item was made available to display in tests, the video.

The log was also improved allowing to keep track of the test results in a more accurate way: how much time the participant takes to perform a task, information about of which elements were selected and what time and classification score.

The main goals didn't change much and aim to identify viable interaction methods (gestures, speech commands) of novel and traditional UI paradigms for the different impairments in the target groups using a multimodal system in realistic scenarios and collect extensive data about users in order to group types of users into clusters. The following sections explain these trials setup, tests and the analysis over the findings.

3.5.1 Setup

Twenty elderly people participated in these trials: five men and fifteen women. Their average age was 68.5 years old, with the youngest being 55 years old and the oldest being 84 years old. Sessions lasted about 60 minutes, in which participants familiarized themselves with the available means of interaction and executed the required tasks. The measurements taken in these trials were the number of errors, time to finish a task and the observation of the participants' actions.

The system was installed in a PC with two screens: one regular monitor for the ”wizard” and one TV for the user. The devices available for interaction in these second trials were the WiiMote, Microsoft Kinect and a Remote Control. Also, speech commands were accept as input. The iPad was tested but independently from the system.

Everything was previously setup in one computer in the Faculty of Sciences in Lisbon, where it was tested out by the local team involved in the project, and only then was sent to Ingema in Spain to make sure the system worked as it was supposed.

Also, one person from Ingema was in Germany at IGD to learn the new features of the user test trials application. With this it was made sure that with the knowledge of the system obtained in the first trials and perceiving the modifications made on the several interaction modules, learning the system would be a very short step before beginning with the main study in Spain.

Concerning the technical modifications in the user initialization application, as men-

tioned in previous section, there is one major change related with the substitution of the Led Camera Motion Sensor with the Microsoft's Kinect Sensor, which gave a lot more precision to pointing interaction. Also: the Wii remote interaction was improved to give a better precision to the user experience; the Avatar engine was improved to give a more friendly interaction to the user; it was added visual feedback to voice interaction; and finally the test script was modified so that original tests that were excluded from the previous trials could be integrated in this second trials.

Type of tests	Tasks	Devices/Modalities
Modalities and devices experimentation	Answering to questions related with preferences of interaction, experimentation of each device and modality. Menu items selection and navigation.	Input: Remote control, WiiMote, Kinect. Output: visual menus and Avatar.
Visual capabilities and preferences	Answering to questions related with interface visual configuration (font size and color,etc). Menu item selection and navigation.	Input: one or more devices choosen by the user. Output: visual menus.
Audio capabilities and preferences	Answering to questions related with audio preferences (audio volume).	Input: Speech. Output: Audio.
Cognitive capabilities	Localization of different items on the screen (cognitive scientific tests), measuring time of response.	Input: Speech, WiiMote, Motion Sensor. Output: Visual menus and pictures
Motor capabilities and preferences	Performing gestures. Menu item selection and navigation. Interacting with the tablet, entering numbers, making gestures and interacting with a map. Answering to questions related with motor preferences and pointing mechanisms.	Input: WiiMote, Kinect and tablet. Output: Visual menus.
Avatar preferences	Interacting with avatar. answering to questions related with avatar preferences.	Input: One or more devices chosen by the user. Output:Visual menus and Avatar.
Multimodal preferences	Menu item selection and navigation. Simulation of application contexts of use. Answering to questions related with multimodal interaction and references. Target selection in different positions in the room.	Input:One or more devices chosen by the user. Output: Visual and audio menus and avatar.

Table 3.3: Tests, tasks and modalities

3.5.2 Tests implemented

The tasks the users performed and also the questions about their preferences covered six categories: visual, audio, motor, cognitive, avatar and modality requirements (see table 3.3). These tasks were essentially the same described in the previous section about the first trials. However, some changes were made:

In the first trials only one question was made to choose the most adequate volume of the voice. In these trials two more tests were added, the ability to hear messages at different volumes with the TV sound on and the preferred volume, and preference between female and male voices.

Participants had to perform some additionally tasks in the tablet, such as entering numbers with a virtual keyboard, make different gestures (circular, swipe, pinch gestures) and drag and manage a map using those gestures.

Another new task consisted in the participant pointing at a concrete target on the screen from different positions (close versus distant; sited versus standing; front to the screen versus side to the screen) and then say from which position was pointing most comfortable.

Improvements in the avatar were made and now the participants were asked to compare it with text output and then with audio. Also, three pictures of the avatar were shown: head only, upper body and whole body and the participants were required to express their favourite image.

3.6 Analysis

This section describes the analysis made over the results obtained in both trial based on the participants expressed opinions but also on supervisors' observations, in situ and of trials' recordings. The presented results only relate to interesting findings about user-system dialogue variations. For more detailed results consult [10] and [7].

3.6.1 Specific Modality and Multimodal patterns

Most participants employed multiple modalities in a redundant fashion, speaking the option's text and pointing to it. From the interaction analysis it was possible to identify some integration patterns. Some participants pointed before speaking, while other participants spoke before pointing. A few participants combined the pointing with spoken deictic expressions. In one of the trials the participant used deictic reference while pointing, and then followed it speaking the option's text, introducing a different interaction pattern.

Other interesting observations regarding the use of multiple modalities were made. Some participants, even in tasks where they were required to select only by pointing

ended up also speaking the option's text, even without noticing it. In one case, even after being asked to point the correct option, a participant just spoke the option's text.

In what concerns the use of speech alone for selection operations it is important to understand if users simply read aloud one of the choices presented to them or if they use alternative speech. Although most participants simply read aloud one of the presented options, we also witnessed some natural utterances, like saying "the fourth" instead of "option 4" which was the text presented on screen.

These findings confirm the need of Dialogue Manager to generate a set of words that will identify the current state elements the user intends to select, not only taking in account the element name but also its order or position. When interacting with a GUIDE application or with the system itself, a set of speech commands should be recognized every time the user says them. Words like "Select", "This", "Yes", "No" and "Confirm", should be included in a speech recognizer dictionary, maintained and updated by the DM regarding the application context, with the objective of helping in the selection of a button or option to where the user is pointing when the command is issued. The system should also recognize the number or order for selecting buttons presented on the screen. "One", "Two", "Three", "First", "Second", "Third", etc., should represent keywords, supporting redundancy in the selection when a user is pointing to an element on the screen, or used alone when the user is interacting with using only speech. Additionally, it is clear that exists the necessity to enable the system to adapt the interaction with the application regarding single and combined modalities.

Regarding the preferred modality for output, most people wanted to receive the messages given by the system only by audio. However when introduced different contexts: watching TV or browsing images, the preferences change. In the first context there was not a consensus but the majority preferred to receive them by means of a text shown on the screen and read by the avatar at the same time. However, this was not the preferred modality when they were browsing the images. In this situation they prefer the text shown on the screen but read for an artificial voice. According to what the users reported, they preferred the avatar when they are watching TV because this catch up their attention while in the context of browsing photos they are not so absorbed so they can more easily switch the focus of their attention. This means that preferences change not only through time but also through different contexts of use. Therefore, the Dialogue Manager must take in account these variables in order to assist the Fission module in the adaptation process, and update the knowledge inside User Model.

User interfaces with a lot of buttons, or small buttons (or both) are difficult to interact without any pointing filter, therefore, selection of items/buttons should be made by target approximation or making use of interaction filters. Also, for users with movement impairments or difficulties, GUIDE should be capable of recognizing difficulties in pointing interaction (for example by measuring long periods of screen navigation without any

selection occurring) and adapt the interaction by applying interaction filters. This means that the Dialogue Manager needs to maintain a representation of the interactive elements in order to assist the component responsible to apply these interaction adaptations.

3.6.2 Cognitive capabilities

Regarding cognitive capabilities two tests were made. One to assess the participants' visual memory, where it was presented different objects in the screen and the participant had to recall their position. The other test evaluates the attention memory, where this time the participant had to focus his attention on one of the objects and recall its position.

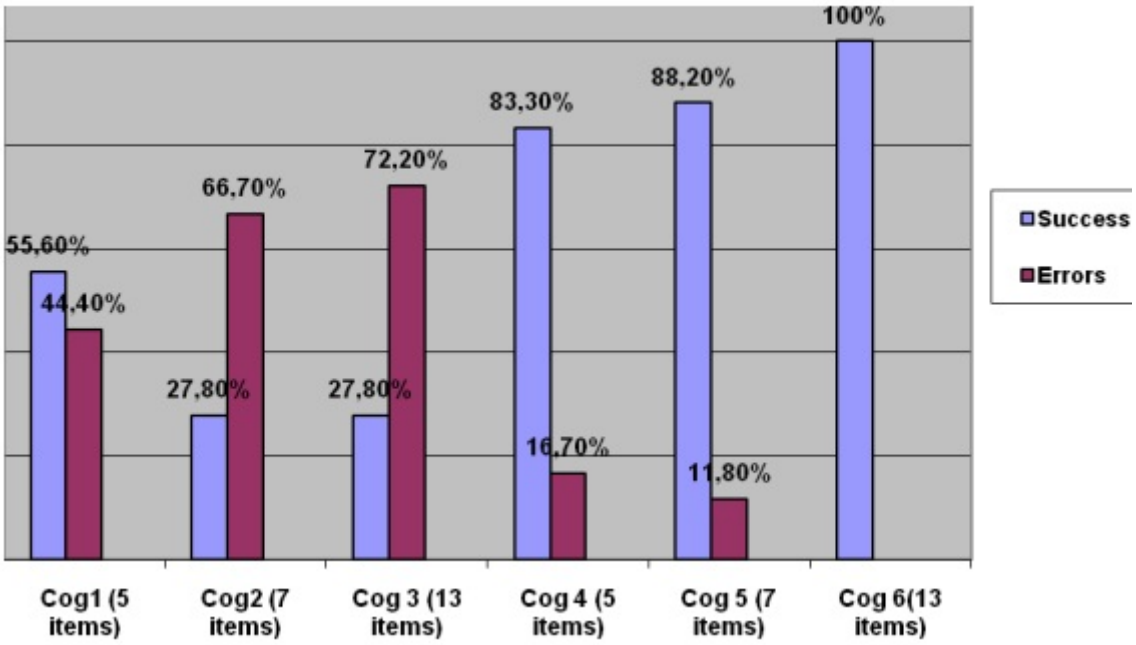

Figure 3.4: Percentage of success in the cognitive tests

As it can be seen in Figure 3.4 the percentage of errors is higher on those questions assessing visual memory (Cog1, Cog2, Cog3) than in those which evaluate the visual attention (Cog4, Cog5, Cog6). This result is congruent with the ones found in literature about the cognitive status in the elderly. In the visual memory questions the more items were required to memorize the higher the error rate was. However, in the visual attentional questions the pattern was the inverse one. These results acknowledge the fact that cognitive load of the user interface must be taken in account in order to ease the interaction. A solution would be that Dialogue Manager could divide a state in different screens with less information and interactive elements each and therefore warn the Fusion module that less commands are expected.

3.6.3 Disabilities

Participants with visual impairments preferred the biggest button and text sizes, regarding buttons placement they opted for 8 buttons with large inter-spacing. Additionally, they preferred the use of modalities redundantly both for input (e.g. pointing and speech) and output (e.g. text and audio). Also, as expected, they preferred the use of the avatar (whole body) over plain text. However, there wasn't a consensus in some relevant variables such as preferred device for interaction, central or peripheral location of the buttons, text colors, etc.

Participants with hearing disabilities preferred the highest volumes on both scenarios (TV off and on) and the audio modality to always be complemented with visual information. Curiously they preferred also the avatar over the use of text. Also, it wasn't found a consensus in some relevant variables such as preference about female or male voices and avatar versus TTS.

There was not a tendency in preferences regarding people with cognitive or motor impairments. However, an agreement can be found in the preferred way of interaction since most of them opted for speech commands.

3.7 Discussion

This chapter has shown the importance of the user trials and their results for a user centred system such as GUIDE. Trials showed that each user has its own characteristics and interaction patterns, and thus, the confirmation of the importance of implementing an adaptive multimodal system.

Even though GUIDE possesses different components that apply adaptation techniques, the Dialogue Manager takes the central role linking those components, using them as resources in the adaptation process. Besides, the DM maintains not only a representation of the dialogue between the user and the application but also with specific GUIDE functions (e.g. raise the volume, help system) that can be actioned by the user.

These trials also revealed the importance of maintaining an User Model, thus, somehow information about the user must be collected by the system in order to perform a satisfactory adaptation. As referred before, one of the goals of these trials was to come up with a series of identifiable clusters. [7] describes those clusters and which characteristics define each one. Thus, arose the need to build an application capable of tutoring the user on how to interact with the system and also gathering the user's abilities and preferences in order to match the user to a certain cluster. The result was the User Initialization Application which is described in the next chapter.

Chapter 4

User Initialisation Application

The User Initialisation Application (UIA) is a prototype for a specific application that will run on the GUIDE platform to start the user interaction process and detect user profiles of users with special requirements. So, when a user starts using GUIDE the User Initialisation Application is the first thing he has to go through, and the first step for making adaptation possible.

4.1 Introduction

The UIA resulted from the need to share knowledge about themselves between user and system. GUIDE needs to know what are the user abilities to be able to adapt interaction to its users. The user needs to know what GUIDE allows him or her to do interaction wise. These two requirements converged in a single application. This application is presented to the user as a tutorial, informing what can be done in the scope of GUIDE and how it can be done.

The UIA presents each modality that can be used to interact and then instructs the user to do simple tasks such as selecting a button with the described modality, as figure 4.1 shows. Then combinations of modalities are introduced, explaining that the user can point and say ”select” and that produces the same result as using the navigational keys and select button of the WiiMote for example.

Then users are asked to repeat instructions shown and are focused on physical, sensorial and cognitive capabilities. They are also asked to identify their preferences regarding font size and colors, for instance. From the user point of view, they are practising and understanding how known interaction modalities (speech and gestures) can be used to operate TV based applications, and they are configuring their environment according to their preferences. At the same time, GUIDE is collecting the data it needs to be able to understand the abilities of the user.

GUIDE users start by belonging to a cluster that defines their abilities. These clusters resulted from the scores of user trials conducted and described in the previous chapter.

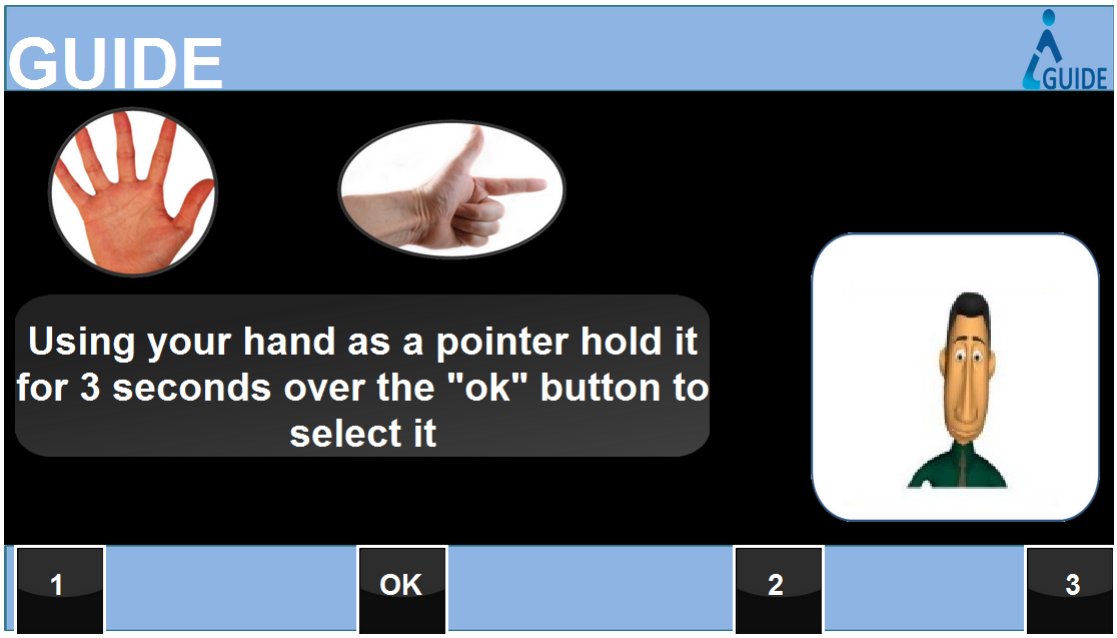

Figure 4.1: User Initialisation Application tutoring user on how to use the hand to select

The goal of the UIA is, by the end of its execution, to be able to identify the cluster the user belongs to. Clusters are defined for visual, audio, motor and cognitive skills. Each skill is characterized by a set of defining characteristics. The UIA gathers information about the users to be able to discern the likely value of that characteristic for that user.

The information collected from all tasks and selections performed by the user is thus used by the User Model component to identify the cluster the user belongs to. This will enable a good initial fit, which means the user experience with GUIDE can be adapted right from the beginning, towards a more usable and satisfying interaction with the system. Afterwards, while using other applications, the interaction is monitored and the defining characteristics' values are fine-tuned, so that the user experience can improve constantly.

4.2 Architecture

The UIA pretends to be a regular web based application as any other that will run on GUIDE's environment. However, because GUIDE is not yet fully implemented this prototype runs on a Java basic version of the system. This prototype is a web based application (HTML and Javascript) that uses an applet (Web Browser Interface in figure 4.2) to perform the communication between the UIA and this simple "GUIDE core". All the actions performed in the application are sent to the "GUIDE core", for instance the preferences collected by the UIA are received by "GUIDE core" which uses it to perform changes in the user interface (changing the CSS file).

The Fusion is simplistic and implemented in Javascript outside the Java implementa-

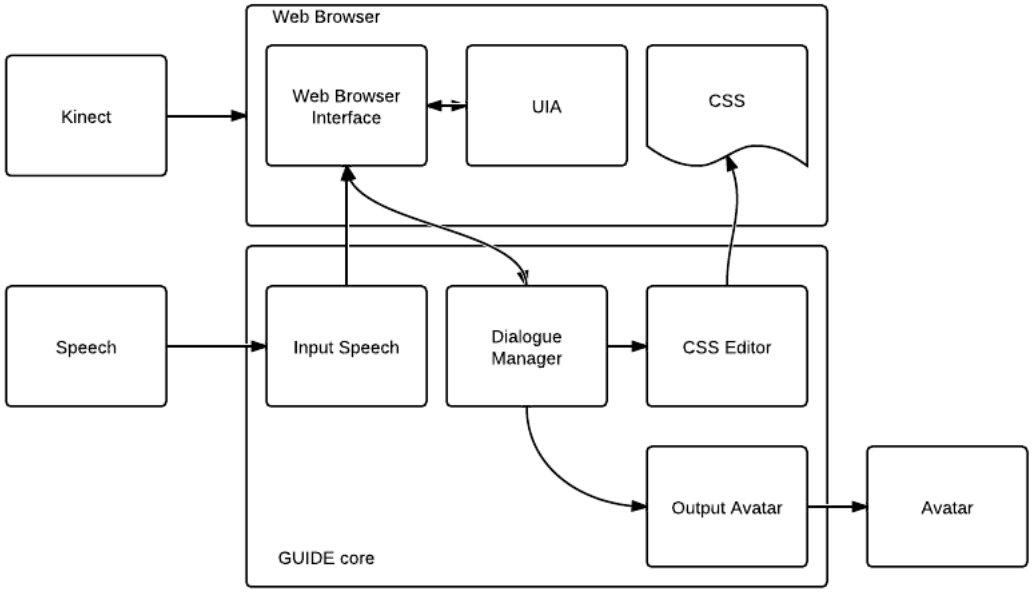

Figure 4.2: Architecture of the User Initialisation prototype

tion. We opted for this since the events and element information were intercepted by this Javascript component, as at this time we haven't got a standard format to represent the User Interface in the system.

The Dialogue Manager implemented is also basic and it's main goal is to know which state is to be presented next, in this case a simple list of HTML pages which are loaded in order, and deliver other messages to the other components such as the avatar and CSS editor.

The Fission Module is composed by the avatar communication component and a CSS editor that adapts the user interface according to the information received in run time. The application communicates the choices of the user to system using API functions offered by the WBI.

4.3 Tests implemented

The tasks and questions take into account the characteristics and measurements needed to classify the user. Regarding visual skills the close and distant abilities to read, the general eyesight, night vision and color perception are evaluated. Hearing capabilities are assessed by the user's audio perception in different frequencies (500Hz and 2Khz) and with noisy background. Cognition tasks measures the user's cognitive executive functions while motor skills are evaluated by a mobility diagnose (e.g. hernia / slipped disc), frequency of muscular weakness, difficulty in writing and rigidity of muscles.

This first prototype implements some of the relevant tasks that evaluate users' abilities, however it doesn't mean that those are the ones chosen to be in the final version neither

that's the way they will be presented. One major concern that wasn't taken in account in this prototype is that UIA must not make the user feel he or she is being tested or in the "doctor's appointment".

Before the tasks really begin, the user is presented with a tutorial on how to interact with the application, i.e. the user is informed about the possibility of using his hands to point at the screen and holding on a button for 3 seconds to press it, he can also use the speech to perform actions such as selecting buttons or navigate in the menu, a combination of the two is also permitted (e.g. pointing at a button and say "select"). The remote control is also presented however it was not yet implemented in this version.

4.3.1 Visual

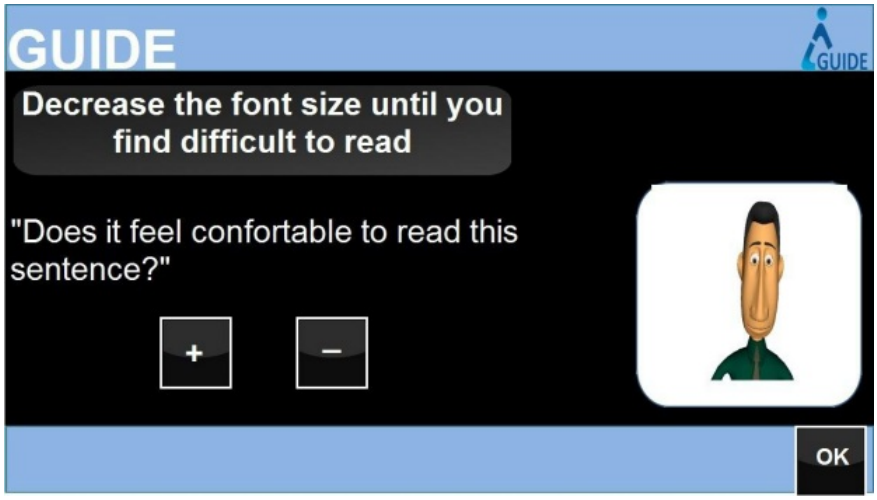

Figure 4.3: Visual perception task on UIA

The first task related with visual perception asks the user to adjust the size of a text presented (see figure 4.3) until he or she finds it difficult to read. This task aim to find the range of font size that user is able to read. Following, several combinations of text and background colors for user to choose which one is more visible are presented.

In what concerns button presentation, the first task asks the user to define the spacing between buttons horizontally and vertically that the user feels acceptable to be able to navigate and select. Next, different combinations of text and background colors for buttons are presented.

4.3.2 Audio

Related to audio perception two tasks are asked to be performed. In the first one the avatar will read some sentences each time the user presses a button. For each sentence read the volume decreases. User stops when he or she can't hear the voice any more. In second

Figure 4.4: Audio perception task on UIA

one the task is similar but with background noise. This aims to gather the volume range in which the user feels comfortable. See figure 4.4 for this task screen.

4.3.3 Motor

Concerning motor skills, the application asks the user to use his left arm to point at a target on the top right side of the screen, and then the inverse. After each task the UIA asks the user if the arm is tired and if it hurts.

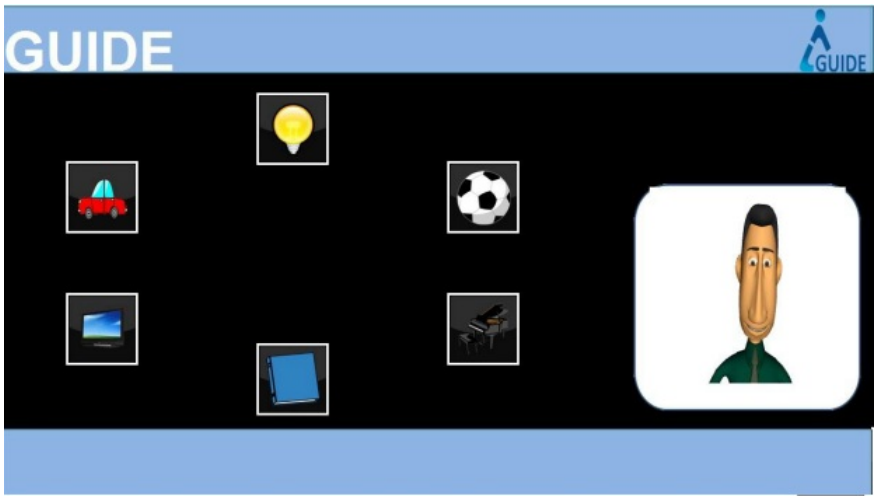

Figure 4.5: Cognitive skill task on UIA

4.3.4 Cognitive

This task (figure 4.5) consists in a game where six images are presented on the screen representing objects that disappear after a short time period. The user is then asked to

locate a random object.

4.4 Discussion

The User Initialisation Application is a key feature on GUIDE to explicitly gather information about the user, as it is the main driver for the adaptation mechanisms included in this project. The information collected is stored in a User Model that will be maintained and updated by the GUIDE core components. This initial prototype will lead to new improved versions with new and enhanced tests, more entertaining and less intrusive.

Chapter 5

GUIDE's Dialogue Manager

5.1 Architecture

The GUIDE Core contains the components that, together with the interface and the application execution environment, support the multimodal interaction and adaptation process. Figure 5.1 presents a conceptual view of the Core and how it relates to the application's execution environment and the input and output devices that are responsible for acquiring commands from the user and conveying to the user the application's content.

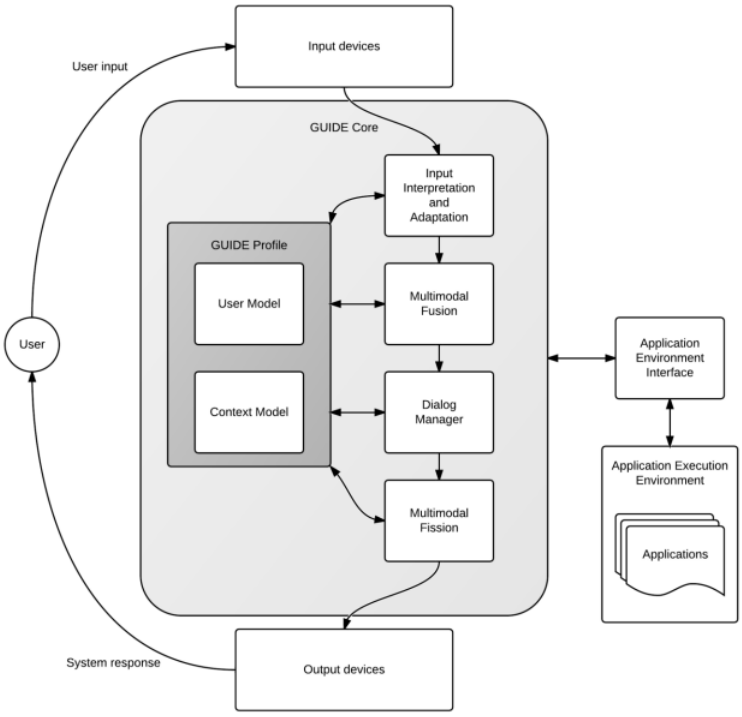

Figure 5.1: Conceptual diagram of GUIDE core

The different components comprising the Core are described in the following subsection. In the remainder of this section, the information flow during an execution cycle is

described.

5.1.1 GUIDE Core Components

Input Adaptation

The input adaptation module facilitates pointing in electronic interfaces by users with motor impairment. It attracts the pointer to the middle of a button, if the pointer is in vicinity of the button. So even if the user points towards the edge of a button, the pointer will automatically move to the centre of the button.

Multimodal Adaptive Fusion

In any system that supports multiple modalities for input, such as gestures or speech, there is a need for a multimodal fusion module. This component of the system is responsible for receiving the incoming input from different sources; combine that information according to specific context or user sensitive- information and making an interpretation out of it.

The main task of the fusion engine of GUIDE is to potentially combine any incoming input from the recognizers, make an interpretation of that data and forward it to the dialog manager. The key to provide the most suitable interpretation is to take into account critical information that is provided by three main sources: the user model, context model and input events.

Dialogue Manager

The Dialogue Manager (DM) coordinates the activity of several components of the GUIDE Core and its main goal is to maintain a representation of the current state of the on-going dialogue the user maintains with the application or with GUIDE itself for changing some interaction preferences. To this end, the DM receives the interpreted commands from the fusion component and reacts accordingly. It has the responsibility to decide when and which state of the dialogue can be triggered. This particularly component is described in more detail further ahead in this chapter.

Multimodal Adaptive Fission

A multimodal adaptive system should be able to flexibly generate various presentations for the same information content in order to meet the individual user's requirements, environmental context, type of task and hardware limitations. Adapting the system to combine all this time changing elements is a delicate task. The fission module is crucial to make that possible as it takes advantage of multi modalities to overcome sensory impairments that users may have.

This core component of GUIDE is responsible for choosing the output to be presented to the user and how that output is channelled and coordinated throughout the different

available output channels. To do this, the fission engine follows three tasks: message construction; modality selection; output coordination.

User Model

The User Model in GUIDE stores the information about the user's characteristics, behaviours and preferences. This data is then used to change and adapt parameters in the fusion and fission components in GUIDE core. For instance, the fission module can ask the User Model to give a range of recommended values for UI elements properties corresponding to the user characteristics.

This information about the user is gathered initially by GUIDE when it recognises a new user to the system. But after determining the initial user's profile, GUIDE will keep updating the user model to better adapt to its user. The most challenging aspect of this adaptation is how to learn the users' characteristics in a way that can be helpful to them. The adaptation, or the user model updating, will take place after explicit user actions, like, for instance, asking to raise the volume, or asking for larger font sizes. However, GUIDE wishes to go beyond adapting to explicit user instructions, and look for other opportunities to improve interaction with the applications.

Context Model

The context model in the GUIDE Core provides a facility for processing and storing contextual information from the different context sources: User context, Application context, UI components, Hardware context. In what concerns User context we can have, for instance, a change in the user's preference for GUIDE to render application content in a different modality, or to increase the volume due to noise produced by a fan, or GUIDE detects that another user entered the living room.

The Application context results of some dynamic behaviour in the application's logic, e.g. a pop- up, an animation, or some unexpected behaviour such a crash.

All UI input components interact with the GUIDE Core using context events that wrap information representing the input made by the user via some input modality (gesture via pointing device, button signals via a remote control or through speech input) or context information about state of the underlying hardware platform/STB. An example could be that the STB lost TV signal reception, or internet connection, or a new Display has been connected to the STB, etc.

5.1.2 Information flow

For user interaction with the system to be possible, an initial rendering has to be presented. The application is responsible for starting this process, and prepares the initial rendering. This happens at the start of "Application time" (see figure 5.2), while GUIDE

waits for the application to communicate what is the content that should be rendered. The application communicates this content to the Application Environment Interface. The UI Representation Builder component, residing inside the Application Environment Interface, parses the content and translates it to UIML, the UI representation language that the GUIDE Core understands and forwards the UIML to the GUIDE Core, while delaying the rendering of the content until it receives the adaptation processing results from the Core. This marks the ending of "Application time" and the beginning of "GUIDE time".

This is the period that the GUIDE Core processes raw user inputs and interprets them into a meaningful application command that is then forwarded to the application. The GUIDE Core can be interrupted by the application during "GUIDE Time", thus enabling, for instance, applications to warn users for some event that might happen during the interaction period.

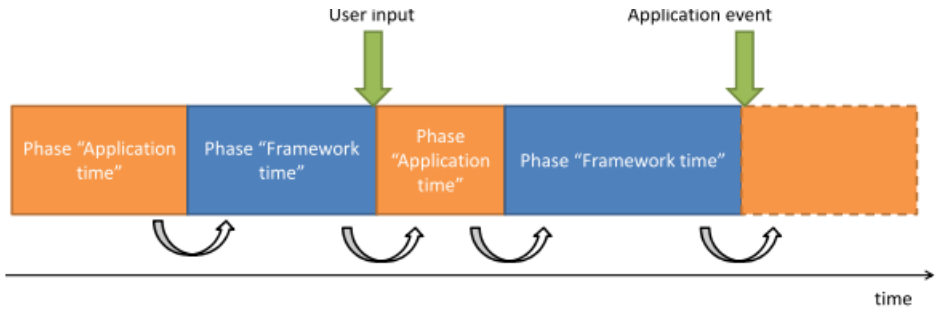

Figure 5.2: Conceptual diagram of GUIDE core

Inside the Core, the Dialog Manager (DM) is listening for the UI representation. After receiving it, the UIML is parsed, looking for all the possible transitions from this application state (e.g. a Web page) to a new application state (e.g. a link to a new Web page). For each transition found, the DM instantiates a frame and listens for inputs from the user that might fill the frame, and thus trigger the transition to a new application state.

At the same time, the User Model also receives the UIML representation of the application interface. Processing this representation, the User Model suggests a range of possible presentation changes for each component in the UI (e.g. the range of suitable font sizes for each button based on the user visual abilities). This initial adaptation of the presentation is then disseminated inside the Core.

The Multimodal Fission component receives the adapted presentation. It parses it, identifying which content is to be rendered to the user. Using information made available by the User Model and the Context Model it decides on alternative or complementary content presentation (e.g. decides to present the buttons text using speech synthesis). It also adapts the UI rendering based on information from the Context Model (e.g. output devices' capabilities or environmental conditions) and the range of possibilities offered by the User Model. It then instructs each output device with the information it should

render, and sends to the Application Environment Interface the adapted UI that should be rendered in the Application Execution Environment.

While this is taking place, the DM sends to the Multimodal Fusion component a list with all the interactive elements that exist in the current application state and its parameters. The Fusion component receives this list and consults the adapted UIML description of the interface to collect required information for each of the elements (e.g. where it is on screen, how big it is, what text it displays, and so on). For each of these elements, Fusion prepares a set of triggers. These are used to interpret user inputs (e.g. one trigger fires when the user points for more than 2 seconds at a button; other trigger fires when the user speaks the button's displayed text; other trigger fires when the user points at the button and speaks "select"). Fusion then sends to Input Adaptation a list of interactive elements currently being rendered on screen (required for applying cursor smoothing techniques and estimate user intent), and for the speech recognition engine it sends a list of interpretable commands on screen that the user might speak.

At this point the application state is rendered and inputs from the user can be expected. The input devices send their data to Core (e.g. pointing coordinates and interpreted speech commands). Eventually the user will produce a command which triggers a frame in the Multimodal Fusion component. This is forwarded to the DM who receives the interpreted command and fills a corresponding frame slot. When a DM's frame has been filled a new state should be loaded by the application. For this to happen the Application Environment Interface is notified about the user command and translates it to something the application can interpret (e.g. a click event on a button). Receiving this event, the application will respond accordingly and a new output rendering process is initiated.

5.2 Dialogue Manager

From the several approaches described in chapter 2, it was decided to opt for the frame-based one, given that it matches the project's requirements. This approach is based in form-filling tasks where some sets of information needs to be gathered, and when completely filled an action is triggered. Since we plan to adapt only interaction, the sets of information to gather in a specific application state can be acquired from the application's representation. As such we can create the required frames to manage the dialogue with the current state. Additionally, interaction not directly related to the application content or information processing (e.g. a request to raise the volume, which relates exclusively to interaction) can be identified a priori, which means those frames can also be considered. Another important advantage this approach offers, more so in the context of GUIDE with a STB based framework, is that it is the one requiring the least amount of processing power.

An extension of finite stated-based models, frame-based models, was developed to

overcome the lack of flexibility of the finite state models. The frame-based approach, rather than building a dialogue according to a predetermined sequence of system's utterances, takes the analogy of a form-filling (or slot-filling) task in which a predetermined set of information is to be gathered. This approach allows some degree of mixed-initiative and multiple slot fillings. The frame based approach has several advantages over the finite state-based approach: greater flexibility and the dialogue flow is more efficient and natural.

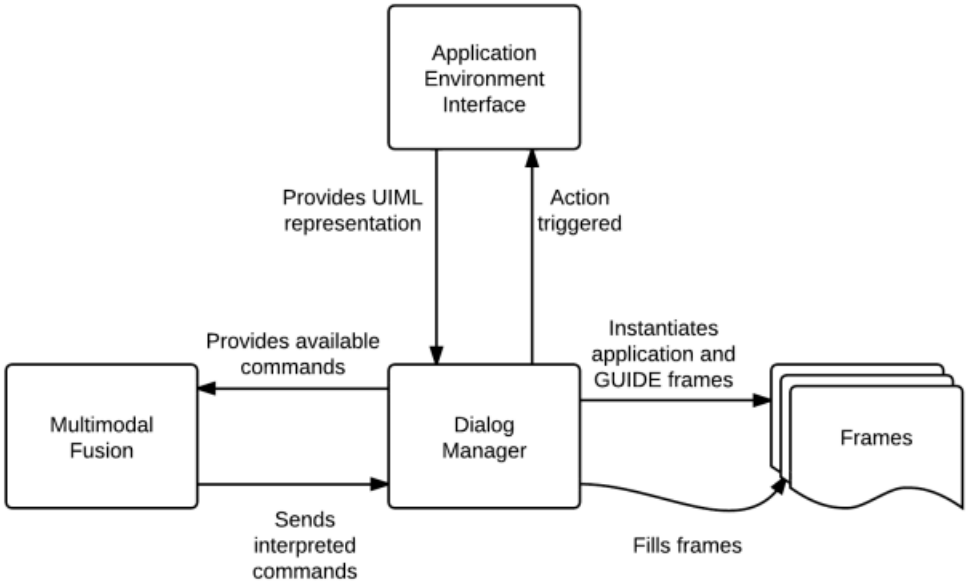

Figure 5.3: Frame-based Dialogue Manager flow of events

As mentioned before, the DM's task is to maintain a representation of the current dialogue. The first step in its operation is processing a representation of the current state of the application (i.e. a specification of what can be seen on the screen). This representation is provided by the Application Environment Interface using User Interface Markup Language (UIML). In the UIML representation the information about all the interactive elements of the current state can be found. In the next step, and because we follow a frame-based approach, the forms must be instantiated (Figure 5.3). To perform that step the UIML must be parsed in order to extract the information needed to trigger each frame. In one state several frames can be instantiated and some of them don't need any slot to be filled besides the click event (e.g. simple links that lead to another state). Also, as GUIDE offers other commands or options which aren't related to the application (e.g. raise volume, increase font-size, help, and so on), some frames are automatically instantiated by the system. Therefore, there are two sources where the DM can derive frames: from the UIML that represents the application's current state and GUIDE native options (Figure 5.4).

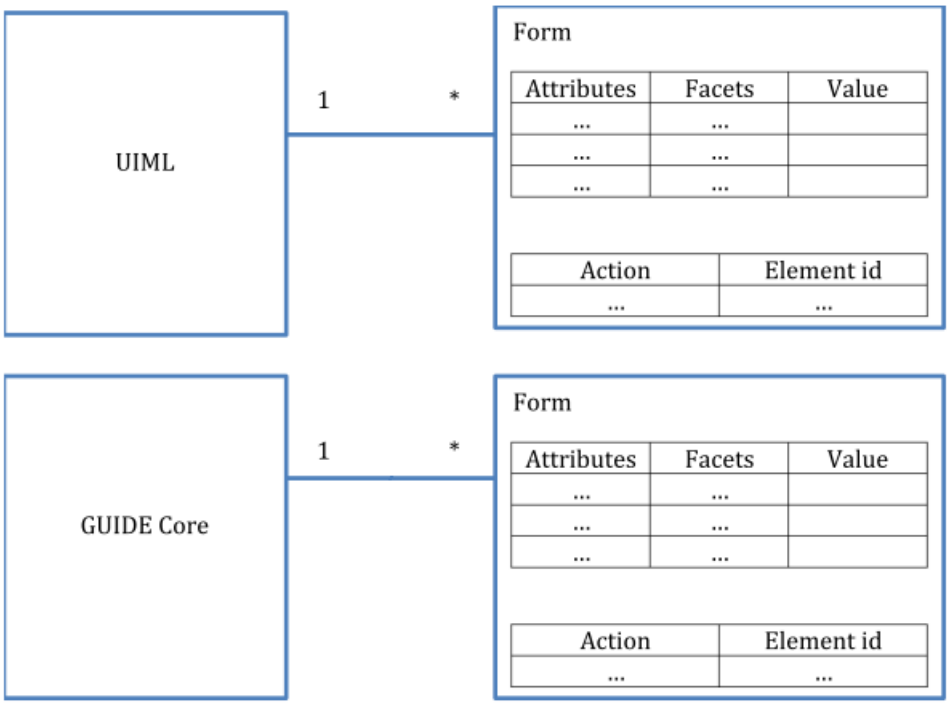

Figure 5.4: Frame instantiation on Dialogue Manager

5.2.1 Frames

The frame is composed of a set of conditions that have to be triggered and the actions that are triggered when the conditions are met. The conditions are specified by a triplet with the interactive element specification (i.e. the attribute), the range of values it can be filled with (i.e. the facets) and a flag to specify if it has been filled or not (i.e. the value). After the frame has been filled it triggers an action (e.g. select, remove, and so on) on an element.

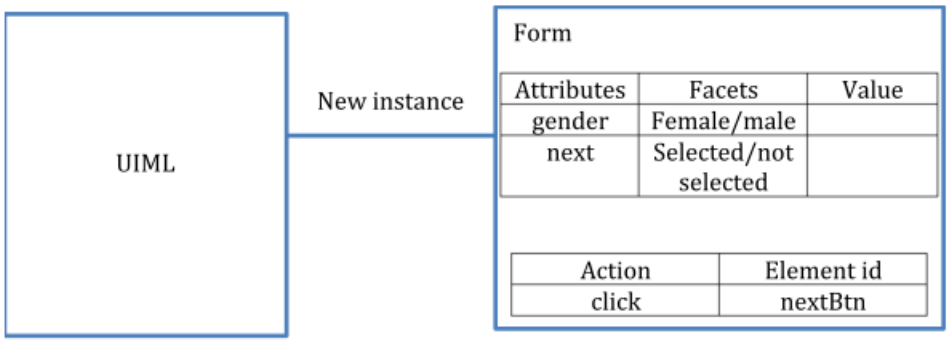

Figure 5.5: Frame filling example

When the frames are instantiated the DM waits for incoming commands from the fusion engine. These commands are associated by the element's id with the interactive elements presented on the screen. For example, if a screen has an option "gender" and

the user has to choose between "male" or "female" and then press a button, the frame would have two slots: gender (facets: male/female) and button "next" (facets: selected/not selected). The action triggered would be the "Click" event (see Figure 5.5). Slots can also be subframes that need to be firstly triggered, taking the same example, the choice of gender can be done using two radio buttons, therefore that selection would need to be communicated to the Application Environment Interface as an "Selection" event.

Most applications, specially when regarding TV, are based in menus that require navigation commands, therefore the Dialogue Manager must take in account those type of interpreted commands coming from the Fusion Module. Thus, directional events should be sent to the AEI (e.g. "Right", "Left", "Up", "Down").

Another type of frames is the GUIDE native frames which are automatically instantiated without need of parsing the UI representation. An example of this would be the "Help System" that is always available to the user regardless of the running application (see Figure 5.6). In such frames the action is not required to impact any specific application element. As such, the corresponding field is left unfilled. The action might not result in any changes to the application (e.g. increase volume) or might impact all application elements indiscriminately (e.g. increase font size).

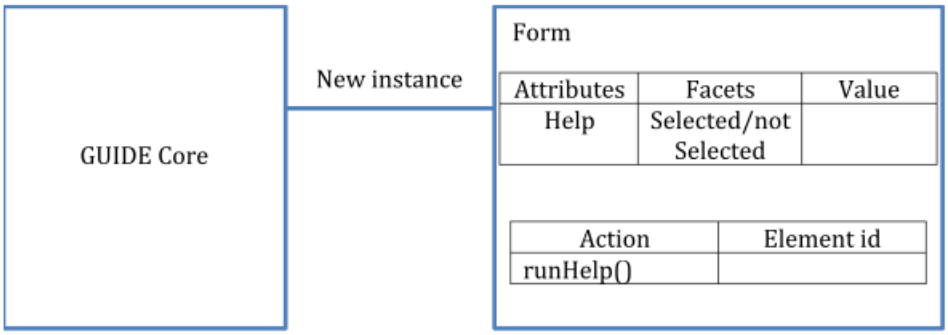

Figure 5.6: Help system frame example

Usually a frame completely filled means a state change and the corresponding action is transmitted to the Application Environment Interface. However, there are some cases where this is not true. For instance, a request to raise the volume is triggered when the related form is filled, but the application's state doesn't change. In this situation, no action is transmitted to the Application Environment Interface. In its operation, besides the aforementioned cooperation with the Application Environment Interface, the DM also needs to cooperate closely with the Fusion component, which is responsible for interpreting the data coming from the different recognizers and communicate that interpretation to the DM. Of course, this data needs to be meaningful to the DM. For that to happen, the DM must provide to the Fusion component information about the interactive elements presented in the current state (including GUIDE native commands). The Fusion component will thus know what to expect from the user and provide only interpretations that are

meaningful to the DM. This important cooperation amongst these two modules is needed for a perfect match between the interactive elements presented in the current state and the commands interpreted by the Fusion engine.

5.2.2 Communication Events

The DM in GUIDE core communicates seamlessly with other components within the core as well as components running outside of the core (e.g. Application Environment Interface) through a bus-based communication framework named GUIDE Baseline User Interface Framework (GBUIF). The Dialogue Manager uses shared data structures defined in the GUIDE Ontology to communicate with the others components, subscribing and publishing in the bus the respective messages. Next, the events that are sent and received by the DM are described.

ContextEvent(IN): The DM subscribes on the context bus (C-Bus) to receive notifications of new context events. Examples of such events include:

1. Context-free input interpreted as a user command by fusion component. An example of a context-free interpreted user command is a voice command (independent of current application state), requesting a feature not available in current application state; e.g. system-related help or functionality offered by another application.
2. The Application related context-Events (e.g. application-logic related state change).

SystemMenu(OUT): The DM manages a system menu data type through which the system can perform one of the following operations:

1. Notify the user of systen related information
2. Respond to user requests for help
3. Provide the user with an option to switch between GUIDE accessible applications

OutputEvent(INOUT): The DM uses this data type to receive notifications of new output events on the output bus (O-Bus). An example of such output event include:

1. An OutputEvent of an active dialogue (application and system) representing the current state and frames to be filled in the current state of the application
2. An update on an OutputEvent of an active dialogue initiating an adaptation of the user interface by the application or change in output modality by fission component

ServiceRequest(INOUT): The DM uses the service bus to directly invoke a change effect in the Application Environment Interface hosting the application. Such effects include:

1. A serviceRequest to the Application Environment Interface (AEI) to load the start page (e.g. a GUIDE MainMenu page)
2. ServiveRequest to AEI load/display a given GUIDE application
3. ServiceRequest to AEI to interrupt/resume/exit from a current application state

5.3 Discussion

This chapter described how the Dialogue Manager will work within the GUIDE core, how the frame-based approach fits in the GUIDE concept, its functions and communication handling. However, a limited dictionary of interpreted commands coming from the Fusion Module as well as the possible actions the Dialogue Manager can send to the AEI must be defined. Was also mentioned that DM makes use of a UI description in order to instantiate the frames, however that description must be delivered by the Application Environment Interface somehow, as it deals with the application directly. Next chapter describes a tool implementation that extracts the user interface from Web based applications.

Chapter 6

User Interface Representation Extraction Component

The Application Environment Interface has the main goal of linking applications and the GUIDE Core and deliver the user interface representation. The main advantage of this approach is to allow any application regardless of its implementation language to communicate with GUIDE in order to adapt its user interface and perform the frame instantiation by the Dialogue Manager. To prove the adequacy and feasibility of this concept, in the scope of the project, one such Interface will be implemented. The Web Browser Interface (WBI) is an implementation of an Application Environment Interface that connects the GUIDE Core to a Web browser where applications are being executed. The WBI is a software component that implements the interface between the GUIDE Framework and a corresponding Web browser application (e.g. HbbTV or HTML 5 based, etc.). The WBI is developed by Fraunhofer, and includes a sub module by the partner FCUL. This chapter describes that module, named User Interface Representation Extraction Component.

6.1 User Interface Markup Language

The basis for communication of an application and the GUIDE Framework is the UI Representation format, that abstracts knowledge about the applications (legacy) user interface and makes it usable by the GUIDE Framework, e.g. to perform adaptation or to initialise the frames for the Dialogue Manager. This representation must contain information about the interactive and visual elements of the interface, such as background colour, button position or text size. This approach has the advantage to use the GUIDE core in different environments without any internal change, plus not limiting application development to a single programming language.

Being a component of the WBI, the UI Representation Extraction Component has the goal of extracting the UI representation from the HTML code. For other execution environments similar UI Representation Extraction components have to be developed.

Several abstract and concrete user interface markup languages were taken into account to be used as the UI standard representation for GUIDE:

XForms (http://www.w3.org/MarkUp/Forms/) is an XML application that represents the next generation of forms for the Web, and has introduced the use of abstractions to address new heterogeneous environments. When comparing XForms with HTML Forms, the main difference, apart from XForms being in XML, is the separation of the data, from the markup of the controls. This not only makes XForms more tractable, by making it clear what is being submitted and where, but it also eases reuse of forms, since the underlying essential part of a Form is no longer bound to the page it is used in. A second major difference is that XForms, while designed to be integrated into XHTML, is no longer restricted only to be a part of that language, but may be integrated into any suitable markup language. In the XForms approach, forms are comprised of a section that describes what the form does, called the XForms Model, and another section that describes how the form is to be presented.

UsiXML (http://www.usixml.eu/), which stands for USer Interface eXtensible Markup Language, is a XML-compliant markup language that describes the UI for multiple contexts of use such as Character User Interfaces (CUIs), Graphical User Interfaces (GUIs), Auditory User Interfaces, and Multimodal User Interfaces. In other words, interactive applications with different types of interaction techniques, modalities of use, and computing platforms can be described in a way that preserves the design independently from peculiar characteristics of physical computing platform.

XIML (http://www.ximl.org/) is an extensible XML-based specification language for multiple facets of multiple models in a model-based approach, developed by a forum headed by RedWhale software. It was introduced as a solution that enables a framework for the definition and interrelation of interaction data items.

UIML (http://www.uiml.org/), User Interface Markup Language, is an example of a language that has addressed the multi-device interface issue, an XML-compliant language that supports a declarative description of a user interface in a device-independent manner. In UIML, a user interface is a set of interface elements with which the user interacts. These elements may be organized differently for the different categories of users and types of appliances. Each interface element has data (e.g. text, sounds, images) for communicating information to the user, and it also receives information from the user using artefacts (e.g. a scrollable selection list) from the underlying application. Since such artefacts can vary from device to device, the actual mapping (rendering) between an interface element and the associated artefact (widget) is done using a style sheet.

XForms cannot cover all the requirements for GUIDE as it is incapable of describing more specific properties or attributes of certain elements (Buttons, Images, etc.) such as position values, size, colour or other style properties. Although much more complete than XForms, UsiXML fails in giving the location of objects which is an important property

for Fusion and User Model components in the GUIDE Core. Although the tag and property names are a bit sloppy they are indeed complete but the main drawback connected to XIML is the fact that, differently from the majority of the other User Interface description languages, it is developed within a software company, and therefore its use is protected by copyright. UIML specification does not define property names. This is a powerful concept, because it permits UIML to be extensible: one can define whatever property names are appropriate for a particular element of the UI. For example, ”colour” might be a useful property for a button, while ”text-size” might be appropriate for a label. This flexibility allows us to define all the properties needed for all GUIDE components (Input Adaptation, Fusion, Fission, Dialogue Manager, User Model, etc.). Additionally, they might be used to represent the information developers might provide using WAI-ARIA markup tags. UIML seems to be the most complete and flexible UI representation, therefore it was chosen as the standard language to be used inside the GUIDE Core.

Figure 6.1: Example of screen shot from a video-conferencing web based application

As an example of UIML, consider a possible state of a video-conferencing application show in Figure 6.1. An excerpt of its UIML representation is presented in Figure 6.2.

6.2 Previous approaches

Before the UIREC was made part of the WBI different approaches on where and when this extraction was supposed to be done were considered. One of the first approaches was to do it in design time, i.e., when developers finished their application they would use this tool to extract a UIML file describing the entire application. However, this approach was discarded because different developers use different implementation methods, for instance, some developers use separated files for different application states (e.g. one HTML file for each state) and other developers use one single file. The next approach was to integrate this component in the GUIDE core serving as an Application Model that would derive and extract automatically the states and UI representations. However, the

```
<?xml version="1.0"?>
<!DOCTYPE uiml PUBLIC  "-//OASIS//DTD UIML 4.0a Draft//EN" http://uiml.org/dtds/UIML4_0a.dtd>
<uiml>
<peers>
   <presentation base=" HTML_4.01frameset_Harmonia_0.1 "/>
</peers>

 <interface>
   <structure>
     <part class="Div" id="leftMenu">
       <part class="Div" id="callTimer">
             <part class="Font" id="time" />
       </part>
       <part class="Button" id="hangUp" />
     </part>
     <part class="Div" id="rightMenu">
       <part class="Button" id="help" />
       <part class="Button" id="settings" />
       <part class="Button" id="imageBrowser" />
       <part class="Button" id="volume" />
       <part class="Button" id="contacts" />
     </part>
     <part class="Div" id="call" />
     <part class="Video" id="conferenceVideo" />
     <part class="Button" id="fullScreen" />
   </structure>

   <style>
     <property part-name="leftMenu" name="position">[0-1],[0-1]</property>
     <property part-name="leftMenu" name="width">[0-1]</property>
     <property part-name="leftMenu" name="height">[0-1]</property>

     <property part-name="callTimer" name="background-color">[0-255],[0-255],[0-255]</property>
     <property part-name="callTimer" name="position">[0-1],[0-1]</property>

     <property part-name="time" name="text">[string]</property>
     <property part-name="time" name="font-family">[string]</property>
     <property part-name="time" name="text-size">[int]</property>
     <property part-name="time" name="font-color">[0-255],[0-255],[0-255]</property>
     <property part-name="time" name="background-color">[0-255],[0-255],[0-255]</property>

     <property part-name="hangUp" name="text">[string]</property>
     <property part-name="hangUp" name="position">[0-1],[0-1]</property>
     <property part-name="hangUp" name="width">[0-1]</property>
     <property part-name="hangUp" name="height">[0-1]</property>
     <property part-name="hangUp" name="font-family">[string]</property>
     <property part-name="hangUp" name="text-size">[int]</property>
     <property part-name="hangUp" name="font-color">[0-255],[0-255],[0-255]</property>
     <property part-name="hangUp" name="background-color">[0-255],[0-255],[0-255]</property>
           (…)
     </style>
 </interface>
</uiml>
```

Figure 6.2: Sample of UIML representation of the video-conferencing application

same inconveniences appeared and additionally this approach would require a change inside the GUIDE core every time a new application language is required. Finally, the final approach is to make this extraction state by state and outside the core. This approach has the advantage of delegating to the Application Environment Interface the job of delivering the code that belongs to a determined state, and any change in the program language doesn't imply any change in GUIDE's main components.

6.3 Parsing UIML from a Web based application

In the particular instance of the implementation of the Web Browser Interface (WBI) for Web and TV applications, the UI Representation Extraction component (UIREC) must be capable of parsing HTML pages into UIML representations. This requires the parsing of HTML elements to identify which content is presented on page, which interactive elements are available to the user (e.g. buttons) and what are the connections to other

pages. Additionally, the UIREC must also parse the style information of each HTML element in order to provide information on visual rendering of the different elements comprising the page. This will be required for the analysis performed in the User Model to understand the adequacy of visual rendering of each component for the given system's user. This section describes how this prototype, developed in C++ performs the extraction and builds the User Interface Markup Language (UIML) from a web based application, using a simple example. This demonstration will be based on the web page shown in figure 6.3, composed by a menu positioned in the left screen, regular paragraphs and titles, and a form. Style information is in a CSS file as well as in the HTML file.

Figure 6.3: Example of web page that will be parsed using the UIREC prototype

The extraction procedure has two main phases: get the structure of the HTML and get the style from both HTML and CSS. The parsers used belong to htmlcxx, a HTML and CSS API for C++. The structure is obtained parsing the HTML code (a string) and then navigating a tree similar to a DOM tree. UIML needs to classify each element presented on the interface and identify them with a unique id. Therefore, for each element presented in the application there are two properties that need to be specified by the developer: the

id, using the normal HTML id and a class, using some roles described in WAI-ARIA documentation (e.g. menu, form, button, etc.). As can be seen in figure 6.4, every user interface elements have the id and role identified.

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="test.css" />
</head>
<body id="body" role="application">

<div id="leftMenu" role="menu">
<button  id="helpL" role="menuitem">Help</button><br/>
<button  id="settingsL" role="menuitem">Settings</button><br/>
<button  id="imageBrowserL" role="menuitem">Images</button><br/>
<button  id="volumeL" role="menuitem">Volume</button><br/>
<button  id="contactsL" role="menuitem">Contacts</button><br/>
</div>

<div id="info" role="content">
<h1 id="title" role="heading" style="color:blue;">"Neque porro quisquam est qui dolorem ipsum quia
 dolor sit amet, consectetur, adipisci velit..."</h1>
<img id="imgA" role="img" src="awesome.png" alt="Awesome face" />
<p id="text" role="paragraph">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam imperdiet
vestibulum dui, nec rutrum metus rhoncus sed. Nunc id sem at diam vestibulum tempor. Maecenas sed tortor
a lectus placerat ullamcorper. Vestibulum massa eros, pretium quis egestas ut, mollis ac libero. Ut
sodales purus in turpis ullamcorper cursus. Vivamus sagittis viverra ultricies. Donec feugiat velit
eget nulla suscipit sollicitudin. In hac habitasse platea dictumst. Nulla nec pellentesque diam. Nec
nibh adipiscing tempor. Integer euismod euismod fringilla. Sed eleifend tellus eget nunc pellentesque
 varius. Ut lorem velit, porta bibendum dapibus nec, dignissim vel ante. Fusce erat nulla, facilisis
eget molestie eu, dignissim et sem. Sed at nulla quis sapien tristique venenatis in eget nisi.
ellentesque convallis ipsum in augue convallis convallis euismod velit adipiscing. Donec imperdiet
 vehicula consectetur. Phasellus faucibus sollicitudin lectus facilisis tristique.</p>
<a id="alink" role="link" href="index.html" >Phasellus at libero</a><br/>
</div>
```

Figure 6.4: Partial HTML code from the web page example

The style section of the UIML is obtained by parsing both the HTML and the CSS as the properties that describe each element can be found in both places. For instance, the button's text is described in the HTML but the foreground or background color can be found on the CSS. So navigating through all elements it gathers the information presented in the HTML properties, such as "value" or "href", and the corresponding CSS description properties, such as "font-size" or "background-color", which can be found in the CSS file as well in the HTML element property "style". Finished these two phases the UIML is built and returned to the WBI as a string. Figures 6.5 and 6.6 represent partially the output produced by the prototype, the first one shows the user interface structure and the second one the style information of each element.

6.4 Discussion

This chapter described the utility of delivering the application's UI representation making GUIDE's adaptation independent of application's language or environment. It was introduced a component of the Web Browser Interface, a specific implementation of the Application Environment Interface, that handles the conversion needed from HTML to UIML. The initial version of this automatic extraction component is capable of parsing HTML and CSS code and outputting the corresponding UIML representation of the interface. An initial subset of Accessible Rich Internet Applications (WAI-ARIA)tags (mainly

```
<structure>
 <part class="application" id="body" >
 <part class="menu" id="leftMenu" >
   <part class="menuitem" id="helpL" ></part>
   <part class="menuitem" id="settingsL" ></part>
   <part class="menuitem" id="imageBrowserL" ></part>
   <part class="menuitem" id="volumeL" ></part>
   <part class="menuitem" id="contactsL" ></part>
 </part>
 <part class="content" id="info" >
   <part class="heading" id="title" ></part>
   <part class="img" id="imgA" ></part>
   <part class="paragraph" id="text" ></part>
   <part class="link" id="alink" ></part>
 </part>
```

Figure 6.5: Partial structure of the UIML

```
<style>
 <property part-name="body" name="background-color" >#ffffff</property>
 <property part-name="leftMenu" name="background-color" >#ff00ff</property>
 <property part-name="leftMenu" name="float" >left</property>
 <property part-name="leftMenu" name="width" >200</property>
 <property part-name="helpL" name="background-color" >#0000ff</property>
 <property part-name="helpL" name="color" >#ffffff</property>
 <property part-name="helpL" name="font-size" >40</property>
 <property part-name="helpL" name="height" >200</property>
 <property part-name="helpL" name="width" >200</property>
 <property part-name="helpL" name="text" >Help</property>
 <property part-name="settingsL" name="background-color" >#0000ff</property>
 <property part-name="settingsL" name="color" >#ffffff</property>
 <property part-name="settingsL" name="font-size" >40</property>
 <property part-name="settingsL" name="height" >200</property>
 <property part-name="settingsL" name="width" >200</property>
 <property part-name="settingsL" name="text" >Settings</property>
```

Figure 6.6: Partial style information of the UIML

roles described in http://www.w3.org/TR/wai-aria/) has also been selected. The next steps will include further evaluation of the component's performance, by testing it with prototypes of the Web based applications being developed, independently of this tool, inside the scope of the project.

Chapter 7

Conclusions

GUIDE aims to resolve the problems of accessibility in TV based applications by adapting the developed applications' user interface to the users' characteristics and needs and by providing multimodal interaction. While the first feature tries to overcome presentation issues that can restrain users of perceiving the information, the second feature looks up for the interaction problems that a user may have if impaired in a specific modality.

The first prototype implemented, the User Trials Application, not only helped the execution of the user studies in the scope of GUIDE as it also can be used in other projects. This prototype provides the creation of multimodal user interfaces by describing the visual and audio elements in a XML file, proving to be very useful when it is needed to study the users' characteristics and behaviours in early stages of a project.

The user trials are important when implementing a system that is centred in a specific target of the population which is at the same time heterogeneous in terms of preferences and abilities. These studies gave several useful information for the development not only of the Dialogue Manager described in this document but also for the rest of the components such as the Multimodal Fusion module, the User Model and the Multimodal Fission module.

The User Initialisation Application works fine in a simulated GUIDE core, however it has to run as any other TV application. While the GUIDE core and Web Browser Interface are not yet fully implemented, the UIA will continuously suffer more updates on its tasks in order to gather the precise information that GUIDE needs to link the user with the right cluster so that the adaptation is done in the correct way. Additionally, UIA must have a privilege access to the API as it needs to store the information about the user and his preferences.

The design of the Dialogue Manager described in chapter 5 was accepted by all partners. A prototype is being implemented at the moment that instantiates the frames manually not yet by inferring from the UIML description. The interpreted commands will firstly be sent by a console input, simulating the Fusion Module. This prototype will help to understand the commands that need to be specified and which of them are application

or system related. This delay was caused by other delays in the implementation of the GUIDE Baseline UI Framework that is responsible for connecting the different GUIDE core components. Nevertheless, it is expected to be evaluated and integrated with the rest of the components in a few months. It was a goal of this thesis to provide a more complete version of the Dialogue Manager. However due to several non-planned tasks and late decision related to architecture aspects, that was not possible.

The User Interface Extraction Component successfully builds a UIML as output and the integration process with the Web Browser Interface is being done, together with partners from Fraunhofer IGD. However, it will need some adjustments as it wasn't yet tested with a real TV based application. It will be interesting to know if different implementations with different developers impact the correctness of the UIREC parsing. Future work will also deliver a specification of the UIML dictionary in order to define the different elements' classes, properties and values, as GUIDE core components must have a common understanding of the UIML description.

The GUIDE project still has 1 year and 3 moths and we expect that all the goals proposed will be achieved and that the problems of accessibility existent in our society will be at least eased with our contribution.

Glossary

AEI	Application Environment Interface
CCG	Centro de Computação Gráfica
DM	Dialogue Manager
FCUL	Faculdade de Ciências da Universidade de Lisboa
GUI	Graphical User Interface
GUIDE	Gentle User Interfaces for Elderly People
HTML	HyperText Markup Language
UI	User Interface
UIML	User Interface Markup Lanaguage
UIREC	User Interface Representation Extraction Component
WBI	Web Browser Interface
XML	eXtensible Markup Language

Bibliography

[1] Alex M. Andrew and Richard Sutton. Reinforcement learning: An introduction by richard s. sutton and andrew g. barto, adaptive computation and machine learning series, mit press (bradford book). *Robotica*, 17:229–235, March 1999.

[2] L Ardissono, C Gena, P Torasso, F Bellifemine, A Difino, and B Negro. User modeling and recommendation techniques for personalized electronic program guides. *Interface*, 6(2002):3–26, 2004.

[3] David Benyon and Dianne Murray. Applying user modeling to human-computer interaction design. *Artificial Intelligence Review*, 7:199–225, 1993.

[4] Dianne C Berry. The problem of implicit knowledge. *Expert Systems 43 August*, pages 144–150, 1987.

[5] Richard A. Bolt. Put-that-there: Voice and gesture at the graphics interface. *SIGGRAPH Comput. Graph.*, 14:262–270, July 1980.

[6] Trung H Bui. Multimodal dialogue management - state of the art. *Manager*, 2(TR-CTIT-06-01), 2006.

[7] Jose Coelho, Carlos Duarte, Pradipta Biswas, and Pat Langdon. Developing accessible tv applications. *Proceedings of the ACM SIGACCESS Conference in Accessibility (ASSETS)*, 2011.

[8] Daniel Costa and Carlos Duarte. Self-adapting tv based applications. *Proceedings of the 14th International Conference on HumanComputer Interaction HCII*, pages 357–364, 2011.

[9] Carlos Duarte. Design and evaluation of adaptive multimodal systems. *Inform*, 2008. Phd Thesis.

[10] Carlos Duarte, José Coelho, Pedro Feiteira, David Costa, and Daniel Costa. Eliciting interaction requirements for adaptive multimodal tv based applications. *Proceedings of the 14th International Conference on HumanComputer Interaction HCII*, pages 42–50, 2011.

[11] Carlos Duarte, Daniel Costa, David Costa, and Pedro Feiteira. Support for inferring user abilities for multimodal applications. *4ª Conferência Nacional Interacção 2010*, page 2, 2010.

[12] Bruno Dumas, Denis Lalanne, and Rolf Ingold. Description languages for multimodal interaction: a set of guidelines and its illustration with smuiml. *Journal on Multimodal User Interfaces*, 3:237–247, 2010.

[13] Bruno Dumas, Denis Lalanne, and Sharon Oviatt. Human machine interaction. chapter Multimodal Interfaces: A Survey of Principles, Models and Frameworks, pages 3–26. Springer-Verlag, Berlin, Heidelberg, 2009.

[14] Daniele Falavigna, Toni Giorgino, and Roberto Gretter. A frame based spoken dialog system for home care. *Statistics*, pages 2–5, 2005.

[15] Foster. State of the art review: Multimodal fission. 2002. COMIC project Deliverable 6.1.

[16] D.H.W. Hofs, H.J.A. op den Akker, and A. Nijholt. A generic architecture and dialogue model for multimodal interaction. In P. Paggio, K. Jokinen, and A. Jonsson, editors, *Proceedings of the 1st Nordic Symposium on Multimodal Communication*, volume 1, pages 79–91, Copenhagen, September 2003. CST Publication, Center for Sprogteknologi. Imported from HMI.

[17] Michael Johnston, Srinivas Bangalore, Gunaranjan Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn Walker, Steve Whittaker, and Preetam Maloor. Match: an architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 376–383, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[18] Robert Kass. Implicit acquisition of user models in cooperative advisory systems. (MS-CIS-87-05), 1987.

[19] Pat Langley. Machine learning for adaptive user interfaces. In Gerhard Brewka, Christopher Habel, and Bernhard Nebel, editors, *KI-97: Advances in Artificial Intelligence*, volume 1303 of *Lecture Notes in Computer Science*, pages 53–62. Springer Berlin / Heidelberg, 1997.

[20] Ludo Maat and Maja Pantic. Gaze-x: adaptive, affective, multimodal interface for single-user office scenarios. In *Proceedings of the ICMI 2006 and IJCAI 2007 international conference on Artifical intelligence for human computing*, ICMI'06/IJCAI'07, pages 251–271, Berlin, Heidelberg, 2007. Springer-Verlag.

[21] Sharon Oviatt. Advances in robust multimodal interface design. *IEEE Comput. Graph. Appl.*, 23:62–68, September 2003.

[22] Sharon Oviatt. The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications. chapter Multimodal interfaces, pages 286–304. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2003.

[23] Sharon Oviatt, Phil Cohen, Lizhong Wu, John Vergo, Lisbeth Duncan, Bernhard Suhm, Josh Bers, Thomas Holzman, Terry Winograd, James Landay, Jim Larson, and David Ferro. Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions. *Hum.-Comput. Interact.*, 15:263–322, December 2000.

[24] Tim Paek and Roberto Pieraccini. Automating spoken dialogue management design using machine learning: An industry perspective. *Speech Commun.*, 50:716–729, August 2008.

[25] O. Pietquin and T. Dutoit. Aided design of finite-state dialogue management systems. In *Proceedings of the 2003 International Conference on Multimedia and Expo - Volume 3 (ICME '03) - Volume 03*, ICME '03, pages 545–548, Washington, DC, USA, 2003. IEEE Computer Society.

[26] Leah M Reeves, Jennifer Lai, J A Larson, Sharon Oviatt, T S Balaji, Stéphanie Buisine, Penny Collings, Phil Cohen, Ben Kraal, Jean-Claude Martin, and et al. Guidelines for multimodal user interface design. *Communications of the ACM*, 47:57–59, 2004.

[27] M Schneider-Hufschmidt. Human factors (hf): Multimodal interaction, communication and navigation guidelines. *Proceedings of the 19th International Symposium on Human Factors in Telecommunication*, 1:1–53, 2003.

[28] Matthias Schneider-Hufschmidt, Uwe Malinowski, and Thomas Kuhme, editors. *Adaptive User Interfaces: Principles and Practice*. Elsevier Science Inc., New York, NY, USA, 1993.

[29] John R Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969.

[30] David Traum and Staffan Larsson. The information state approach to dialogue management. *Current and New Directions in Discourse and Dialogue*, 4:325–353, 2003.

[31] David Traum and Jeff Rickel. Embodied agents for multi-party dialogue in immersive virtual worlds. In *Proceedings of the first international joint conference on*

Autonomous agents and multiagent systems: part 2, AAMAS '02, pages 766–773, New York, NY, USA, 2002. ACM.

[32] Michael Mctear University and Michael F Mctear. Modelling spoken dialogues with state transition diagrams: experiences with the cslu toolkit. In *Proc 5th International Conference on Spoken Language Processing*, pages 1223–1226, 1998.

[33] Qiaohui Zhang, Atsumi Imamiya, Kentaro Go, and Xiaoyang Mao. A gaze and speech multimodal interface. In *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops - W7: EC (ICDCSW'04) - Volume 7*, ICDCSW '04, pages 208–214, Washington, DC, USA, 2004. IEEE Computer Society.