INTELLIGENT USER INTERFACES

Introduction and survey

Ehlert, Patrick

Research Report DKS03-01 / ICE 01
Version 0.91, February 2003

Mediamatics / Data and Knowledge Systems group
Department of Information Technology and Systems
Delft University of Technology, The Netherlands

Ehlert, Patrick A.M. (P.A.M.Ehlert@its.tudelft.nl)

"Intelligent User Interfaces: Introduction and survey"

Research Report DKS03-01 / ICE 01
Version 0.91, February 2003

Mediamatics / Data and Knowledge Systems group
Department of Information Technology and Systems
Delft University of Technology, The Netherlands
http://www.kbs.twi.tudelft.nl

Keywords: Intelligent user interfaces, adaptive interfaces, human-computer interaction, user modeling, plan recognition, multimodal interaction, intelligent agents, software usability, artificial intelligence

Preface

This report is part of a literature study on intelligent user interfaces. The goal of this report is to introduce the field, explain the concepts and provide an overview of existing (research) intelligent user interface applications.
The target audience is computer scientists or experienced computer users. To fully comprehend the report, some basic knowledge of artificial intelligence is desirable.

Patrick Ehlert,
Delft, February 2003

Summary

Computers are being used for an increasing number of applications and at the same time computer programs are becoming more complex. This justifies the need for better user interface technology. Intelligent user interfaces (IUIs) try to solve human-computer interaction problems by providing new methods of communication and by adapting to the user. Research on new communication methods focuses on natural language systems, gesture recognition, image recognition, and multimodal interfaces. Adapting to the user is done by using techniques from artificial intelligence to perform reasoning and learning, form example user modeling and plan recognition.
Intelligent interface agents, which are anthropomorphic computerized beings, combine several of these techniques. They try to help the user by automating a particular task.
Although undoubtedly, many improvements will be made to the current object-oriented direct-manipulation interfaces that we use now, it is very likely that interface of the future will be very different. Expectations are the adaptive interfaces and multimodal interfaces will be mixed together, creating more powerful and intelligent interfaces.

Used abbreviations

DM	Direct manipulation
HCI	Human-Computer Interaction
IUI	Intelligent User Interface
PBD	Programming By Demonstration
PDA	Personal Digital Assistant

Table of Contents

PREFACE ..... **I**

SUMMARY ..... **III**

CHAPTER 1: INTRODUCTION **1**

1.1 COMPUTERS FOR EVERYONE **1**
1.2 WHY USE INTELLIGENT USER INTERFACES? **1**
1.3 REPORT OVERVIEW **2**

CHAPTER 2: WHAT ARE INTELLIGENT USER INTERFACES? **3**

2.1 DEFINING THE FIELD **3**
2.1.1 PROPERTIES 4
2.1.2 INTELLIGENT USER INTERFACES VERSUS INTELLIGENT SYSTEMS 4
2.2 HISTORIC BACKGROUND **5**
2.3 THE GENERAL DESIGN PROCESS **6**
2.3.1 ANALYSIS 6
2.3.2 DEVELOPMENT AND IMPLEMENTATION 7
2.3.3 EVALUATION 7
2.3.4 REFINEMENT AND TOOLS 8
2.4 OTHER DESIGN METHODS **8**
2.4.1 DESIGN GUIDELINES 8
2.4.2 THE GENERAL ARCHITECTURE 8
2.5 CRITICISM, PROBLEMS AND PROPOSED SOLUTIONS **9**
2.5.1 TRANSPARENCY AND PREDICTABILITY 10
2.5.2 CONTROL 10
2.5.3 MISTAKES 10
2.5.4 FALSE EXPECTATIONS 11
2.5.5 RESPONSIBILITY 11
2.5.6 PRIVACY AND TRUST 11

CHAPTER 3: COMMUNICATING WITH THE USER **13**

3.1 INTRODUCTION **13**
3.2 NATURAL LANGUAGE SYSTEMS **13**
3.2.1 SPEECH RECOGNITION 14
3.2.2 NATURAL LANGUAGE UNDERSTANDING 14
3.2.3 DIALOGUE SYSTEMS 15
APPLICATION 1; THE ALPARON DIALOGUE MANAGER 15
APPLICATION 2: MIXED-INITIATIVE DIALOGUE SYSTEMS 16
3.2.4 DATABASE QUERY 16
3.2.5 SPEECH OUTPUT 16
3.2.6 AN EXAMPLE 16
3.3 GESTURES **17**
3.4 IMAGE RECOGNITION **18**
3.4.1 RECOGNIZING FACES AND FACIAL EXPRESSIONS 18
3.4.2 GAZE TRACKING 18

Application 19
3.4.3 LIP READING 19
Example 19
3.5 MULTIMODAL INTERFACES 20
3.5.1 DATA FUSION 20
3.5.2 APPLICATIONS 20
NATURAL LANGUAGE AND DIRECT MANIPULATION 20
NATURAL LANGUAGE AND GESTURES 21
NATURAL LANGUAGE, GESTURES AND GAZE TRACKING 21

CHAPTER 4: LEARNING FROM THE USER 23

4.1 WHAT IS LEARNING? 23
4.2 KNOWLEDGE REPRESENTATION AND REASONING 23
4.2.1 PREDICATE CALCULUS 24
4.2.2 SEMANTIC NETWORKS 24
4.2.3 FRAMES 24
4.2.4 PRODUCTION SYSTEMS 25
4.2.5 BAYESIAN BELIEF NETWORKS 25
4.2.6 FUZZY SYSTEMS 25
4.3 MACHINE LEARNING TECHNIQUES 26
4.3.1 REINFORCEMENT LEARNING 26
4.3.2 OTHER LEARNING METHODS 26
CASE-BASED LEARNING 26
ARTIFICIAL NEURAL NETWORKS 26
GENETIC ALGORITHMS 27
4.4 USER MODELING 27
4.4.1 CREATING A USER MODEL 27
ASKING THE USER 28
LEARNING USER CHARACTERISTICS 28
4.4.2 APPLICATIONS 28
MICROSOFT'S LUMIERE PROJECT 28
4.5 PLAN RECOGNITION 29
4.5.1 FORMS OF PLAN RECOGNITION 29
4.5.2 APPLICATIONS AND RESEARCH 30

CHAPTER 5: INTELLIGENT INTERFACE AGENTS 31

5.1 WHAT IS AN INTELLIGENT INTERFACE AGENT? 31
5.2 ANTHROPOMORPHIC AGENTS 32
5.3 APPLICATIONS 33
5.3.1 UCEGO; A HELP SYSTEM 33
5.3.2 MOKSAF; USING MULTIPLE AGENTS 33
5.3.3 IBOTS; INTERFACING WITH EXISTING APPLICATIONS 33
5.3.4 OPEN SESAME! 33

CHAPTER 6: FUTURE VISIONS 35

6.1 WHERE DO WE GO FROM HERE? 35
6.2 MULTIMODAL INTERFACES 35
6.3 FINAL REMARKS 36

BIBLIOGRAPHY 37

Chapter 1: Introduction

We will start this chapter with a discussion about the need for good human-computer interfaces in section 1.1. Then in section 1.2, we will provide several reasons how intelligent user interfaces can contribute to solve communication problems. The last section of the chapter (section 1.3) gives an overview of the structure of this report.

1.1 Computers for everyone

If we look at the use of computers in the last fifty years we can distinguish two trends. The first trend is the **increasing use of computers for a growing range of purposes**. Ever since the introduction of the first (electronic) computers in the 1940s, the number of computer users has been growing. Before 1970 computers were regarded mainly as scientific tools and only specialized programmers were able to perform calculations on a computer. With the introduction of the PC in the 1980s this changed drastically. Many people could now afford to buy a computer. In addition, PCs used the (relatively) easy to use command-line and graphical interfaces, so it became much easier to learn how to use a computer. With this a new range of computer applications was developed: word-processors, spread sheets, desktop publishers and computer games. The rise of the Internet and ICT industry during the 1990s further stimulated computer usage [Venkatesh et al 2000]. People with similar interests from all over the world joined in virtual communities and, with the appearance of laptop computers, people were not limited to computer use at home or work but could work anywhere they wanted. Nowadays millions of people are using computers in many different locations and situations. People with a Personal Digital Assistant (PDA) or Internet-capable mobile phone can be connected to anyone, anywhere at anytime. Ubiquitous computing has become a reality!

The second trend that runs parallel to this is the **increasing complexity of computer programs and their interfaces**. With the doubling of computing power every eighteen months predicted by Moore's law, program developers can afford to put more and more functionality into a computer program. If we look at the latest version of Microsoft Word[1] for example, no normal computer user makes use of all its functions, not only because there are so many options and settings, but also because people do not know how to use them or even do not know they exist.

The two developments described above justify the need for good human-computer interfaces. Many computer users are experiencing problems and most of these problems are related to the interface. The encountered problems vary from confusing menu choices and incomprehensible error messages to unnatural (rigid) interaction. Especially beginners, the elderly, or people with disabilities are having trouble, but experienced computer users often bump into problems as well. We need computer interfaces that can understand and help people and explain them how to use the available functions. We need to make sure that computers and other computerized devices remain accessible for everyone.

1.2 Why use intelligent user interfaces?

If we look at the way we interact with computers now, a lot has changed compared to twenty years ago. Instead of the constrained input we used then, interfaces have become much more flexible. Modern-day interfaces try to be intuitive by using the desktop metaphor, which consists of multiple "windows" showing folders and documents (files). However, most modern day interfaces are very limited in handling the differences between users and lack personalization.

Intelligent user interfaces (IUIs) is a subfield of Human-Computer Interaction. The goal of intelligent user interfaces is to improve human-computer interaction by using smart and new technology. This interaction is not limited to a computer (although we will focus on computers in this report), but can also be applied to improve the interface of other computerized machines, for example the television, refrigerator, or mobile phone.

[1] At the time of writing the latest version of Microsoft Word is Word 2002.

Using techniques from artificial intelligence, IUIs deal with different forms of input and output and try to help the user in an intelligent fashion. They try to solve some of the problems that the current direct-manipulation interfaces cannot, such as:

- *Creating personalized systems*
  No two persons are the same and people have different habits, preferences, and working methods. An intelligent interface that takes these differences into account can provide a personalized method of interaction. The interface knows the user and can use that knowledge in the way it communicates with the user.

- *Information overflow or filtering problems*
  Finding the right information on your computer or on the Internet can be like looking for a needle in a haystack. Intelligent interfaces can reduce the information overflow associated with finding information in large databases or complex systems. By filtering out irrelevant information, the interface can reduce the cognitive load on the user. In addition, the IUI can propose new and useful information sources not known to the user.

- *Providing help on using new and complex programs*
  Computer systems can be very complicated to work with when you first start to use them. As you struggle to get to know and understand a new program, new software versions or updates appear including new functionality. Many computer users fail to keep up with these new functions. Intelligent help systems can detect and correct user misconceptions, explain new concepts, and provide information to simplify tasks.

- *Taking over tasks from the user*
  An IUI can also look at what you are doing, understand and recognize your intent, and take over some of your tasks completely, allowing you to focus on other things.

- *Other forms of interaction*
  Currently, the most common interaction devices are the keyboard and the mouse. IUI-research looks at other forms of interaction (e.g. speech or gestures). By providing multiple forms of interaction, people with a disability will be able to use computers more easily.

To summarize, instead of the user adapting to the interface, and IUI can adapt to the user. The IUI tries to determine the needs of an individual user and attempts to maximize the efficiency of the communication with the user.

1.3 Report overview

As mentioned earlier, in this report we will focus on interfaces for computers. We will regard the computer as a tool to perform a certain task. Usually the goal is to maximize job-efficiency and/or improve user satisfaction. We will not discuss games or other leisure uses of computers, although much of our discussion can also be applied there.
In chapter 2 we will look at the definition of an IUI, discuss the design process, and look at the pros and cons of IUIs. Furthermore, we will show that there are two kinds of IUIs. The first type uses new forms of interaction techniques and these techniques are described in chapter 3. Chapter 4 discusses the second kind of IUI that uses adaptation and learning techniques. A special kind of intelligent interface are intelligent interface agents and this is discussed in chapter 5. Finally, in chapter 6 we will look at what the future might bring us regarding computer interfaces.

Chapter 2: What are intelligent user interfaces?

In this chapter we will start with a definition of intelligent user interfaces (IUIs) and describe their properties. Then in section 2.2, we will discuss some of the influences on IUIs and show how the field has developed. Next, we will mention some of the design issues of constructing an IUI (section 2.3) and provide a general architecture (section 2.4). The reminder of the chapter describes some of the criticisms on IUIs and discusses possible solutions (section 2.5).

2.1 Defining the field

As noted earlier, IUIs is a subfield of Human-Computer Interaction (HCI). To make things a bit complicated, in literature the term "intelligent user interface" is used to denote a particular type of interface as well as the research field. Other often mentioned synonyms are adaptive interfaces, multimodal interfaces, or intelligent interface technology. The first two are actually two subtypes of intelligent interfaces whereas the latter is used as a synonym for IUIs as a research field.
The main problem in defining the terms "intelligent user interface" lies in the word "intelligent". For decades, researchers have tried to define intelligence. Back in the 1950s, Alan Turing already came up with a proposal to define intelligence using what we now call the Turing Test [Turing 1950], but the debate still is not settled. Over the years numerous definitions of intelligence have been devised. Most definitions mention the ability to adapt (learn and deal with new situations), the ability to communicate, and the ability to solve problems.
A "normal" **user interface** is defined as a method of communication between a human user and a machine. If we extend this definition, we can say that an *intelligent* user interface uses some kind of intelligent technology to achieve this human-machine communication. In other words, IUIs are interfaces with the ability to adapt to the user, communicate with the user, and solve problems for the user.

> *"Intelligent user interfaces specifically aim to enhance the flexibility, usability, and power of human-computer interaction for all users. In doing so, they exploit knowledge of users, tasks, tools, and content, as well as devices for supporting interaction within differing contexts of use." [Maybury 2001]*

Adaptation and problem solving are important topics addressed by research on artificial intelligence (AI) and therefore many IUIs draw heavily on the techniques developed in AI research. However, not all intelligent user interfaces have learning or problem solving capabilities. Many interfaces that we call intelligent focus on the communication channels between the user and machine. These interfaces often apply new interaction techniques such as speech processing, gaze tracking or facial recognition.
Other research fields have also influenced IUIs. Some examples are: psychology, ergonomics, human factors, cognitive science and social sciences. In Figure 1 we have depicted several IUI research topics and their relationship to the other research fields.

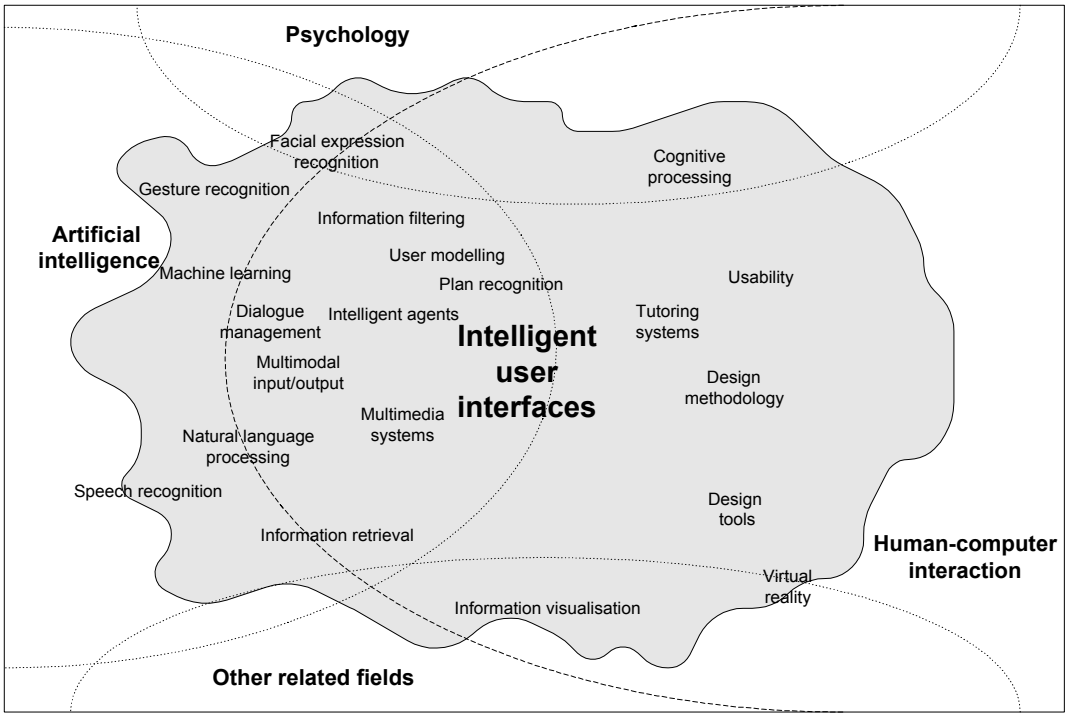

Figure 1: The intelligent user interfaces research field and some of its topics.

2.1.1 Properties

The most important property of IUIs is that they are designed to **improve communication between the user and machine**. It does not matter much what kind of technique is used to achieve this improvement, as long as it can be regarded as "intelligent". Below we give a list of several types of techniques that are being used today in intelligent user interfaces:

- *Intelligent input technology* uses innovative techniques to get input from a user. These techniques include natural language (speech recognition and dialogue systems), gesture tracking and recognition, facial expression recognition, gaze tracking and lip reading;
- *User modeling* covers techniques that allow a system to maintain or infer knowledge about a user based on the received input;
- *User adaptivity* includes all techniques that allow the human-machine interaction to be adapted to different users and different usage situations;
- *Explanation generation* covers all techniques that allow a system to explain its results to a user (e.g. information visualization, or tactile feedback in a virtual reality environment).

Other important properties of IUIs are **personalization** and **flexibility of use**. To achieve personalization, IUIs often include a representation of a user. These user models log data about the user's behavior, knowledge, and abilities. New knowledge about the user can be inferred based on the input and interaction history of the user with the system. In order to be flexible many IUIs use adaptation or learning. Adaptation can occur based on the stored knowledge in a user model or by make new inferences using current input. Learning occurs when stored knowledge is changed to reflect new encountered situations or reflect new data. Because of the difficulties involved in creating IUIs and the amount of knowledge engineering that is needed, most IUIs focus on a specific method of interaction (e.g. speech) or on a particular narrow application domain.

2.1.2 Intelligent user interfaces versus intelligent systems

An often-made mistake is to confuse an IUI with an intelligent system. A system exhibiting some form of intelligence is not necessarily an intelligent interface. There are many intelligent systems

with very simple non-intelligent interfaces and the fact that a system has an intelligent interface does not say anything about the intelligence of the underlying system (see also Figure 2).

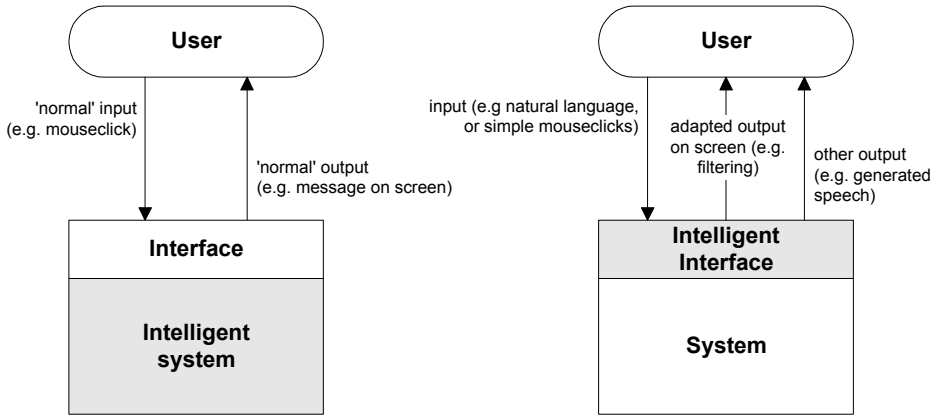

Figure 2: An intelligent system versus an intelligent interface

Unfortunately, the boundary between a system and its interface is not always very clear. Often the technology used in an IUI is also part of the system, or the IUI forms the entire system itself. For example, a speech recognition system can be part of an intelligent interface to a system, but it can also be the complete system. If an IUI can be regarded as a system on its own, then it is by definition an intelligent system. However, an intelligent system does not necessarily have an intelligent interface.

2.2 Historic background

Before 1960, the term "user interface" was practically non-existent. Computer scientists focused on making calculations and the presentation of results hardly received any attention. In the 1960s this started to change. In 1963 Ivan Sutherland published his MIT PhD thesis about a system called Sketchpad [Sutherland 1963]. Sketchpad made it possible to create graphic images directly on a computer screen using a lightpen and was the first graphical user interface (GUI) and direct-manipulation (DM) system.

Around the same time, Douglas Engelbart and William English were working on their "Augmenting human intellect" project [Engelbart and English 1968]. As part of this project they developed a new control device, which was the first mouse. In 1968 Engelbart designed the NLS system; a revolutionary system that was the first to incorporate many things such as hypertext, multiple overlapping windows, elaborate document control, and on-screen video teleconferencing. At the time, the new technologies in the NLS system were really astonishing and groundbreaking. In fact, the NLS system was so far ahead of its time that many people attending Engelbart's first demonstration of the system did not believe it was real.

While in the 1960s researchers focused on new technology, in the 1970s this technology-driven approach slowly changed. Instead, the user started to become the focus of attention and in the 1980s Human-Machine Interaction (HCI) had turned into a user-centered research field with usability as its main goal and technology as a supporting tool. Around 1981, the DM interfaces developed earlier by Sutherland and later by Xerox Parc and Apple, finally were incorporated into commercial software programs. The WIMP (Windows, Icons, Menus, Pointers) model became widespread and proved to be very successful. The WYSIWYG (What You See Is What You Get) concept became a guiding principle for all interfaces.

Although during the late 1980s and 1990s DM interfaces were enhanced with embedded context-menus, new types of mice, joysticks and other controls, the basic technology has not changed much since its introduction. A problem is that new technologies such as data mining, machine learning, speech recognition and computer vision are difficult to use with the existing interfaces.

At the start of the 1990s the field of IUIs slowly started to take form. Around 1994, intelligent agents and recommender systems appeared on Internet. In 1996, the first practical speech

recognition and natural language processing appeared. Then in 1997, Microsoft released their intelligent Office assistant help system. However, since then little IUI applications have been released onto the market.

2.3 The general design process

Common practice in new research fields is that people forget to use design methods and IUIs is no exception. On of the most important design rules is that the intelligent interface should be integrated into the design process from the beginning of a project and not be constructed ad hoc as often happens. Instead of creating an IUI just because "it's cool", the need for an IUI should be founded in the analysis process of an interactive system.
First of all, it should be decided whether a system needs an IUI or not. In general, IUIs are more computationally intensive than "traditional" DM interfaces, so the IUI should provide some added value. If you can get the same results with a DM interface, why bother to create a more complex and costly IUI? The final decision whether or not to create an adaptive mechanism in (commercial) interfaces is one of balancing implementation costs against user-interaction improvement. If some adaptive functionality is implemented it will (if all is well) reduce the cognitive processing needed by the user. On the other hand, an IUI will require time and computer resources to implement and maintain.
The design method that is generally applied in IUIs is that of iterative refinement. It consists of the following steps:

1. Analysis.
2. Development and implementation of (prototype) interface technique or metaphors.
3. Evaluation of the developed system.
4. Make adjustments based on the evaluation results (go back to step 2).

Once this process has iterated to a useful interface technique or metaphor, there can also be a fifth step in the design process:

5. Update interface building tools to incorporate the new technique or metaphor.

In the following sections we will take a closer look at these design steps.

2.3.1 Analysis

The analysis phase is probably the most important phase in any design process, but even more so in IUI design. In the design process of a normal non-intelligent interface one needs to analyze who is the average user, what tasks the interface should support, and on what system they will be performed. With an IUI there is no average user. Ideally, an IUI should be able to adapt to any user in any environment, so the used adaptation technique should be designed in such a way that it can support all types of users. In practice, this is hard to achieve so we simply focus on certain user types. David Benyon [1993] has identified five interrelated analysis activities for designing adaptive systems:

- *Functional analysis* aims to establish the main functions of the system;
- *Data analysis* is concerned with understanding and representing the meaning and structure of data in the application. Data analysis and functional capabilities go hand in hand to describe the information processing capabilities;
- *Task knowledge analysis* focuses on the cognitive characteristics required of users by the system, e.g. the search strategy required, cognitive loading, the assumed mental model, etc. This analysis is device dependent and hence requires some design to have been completed before it can be undertaken;
- *User analysis* determines the scope of the user population which the system is to respond to. It is concerned with obtaining attributes of users that are relevant to the application such

as the required intellectual ability, cognitive processing ability, and prerequisite knowledge required. The anticipated user population will be analyzed and categorized according to aspects of the application derived from task, functional, data and environment analysis;
- *Environment analysis* covers the environments within which the system is to operate. This includes physical aspects of the environment and "softer" features such as the amount and type of user support that is needed.

The result of the analysis phase is a specification of the user's goals and information needs, as well as the system's required functions and information provisions.
A problem that is often encountered in the analysis process of IUIs is the "paradox of change" Since there are hardly any common, functional IUIs, it is difficult to analyze how users will interact with them. On the other hand, if these interfaces are designed and become widely used, one runs the risk that those systems will influence the analysis process. Attempts to overcome this problem usually focus on so-called Wizard of Oz studies. In this kind of study data is collected from a user who is led to believe that he is working on a fully functional and automatic system while in fact the system is being controlled by another person.

2.3.2 Development and implementation

The process of developing new interaction techniques and metaphors is mainly one of creativity. The best way is just to go and try out new concepts and ideas. There are a lot of general guidelines for user-interface design, however most of these guidelines were developed for DM interfaces and are difficult to apply to IUIs. The most important guidelines include [Shneiderman 1992][St. Amant 2000]:

- *Consistency*; a user action under a well-defined set of conditions should always produce the same predictable effect. Obviously, an adaptive interface will violate this rule.
- *Short cuts*; frequent users should be able to use action or command short cuts.
- *Information feedback*; always provide feedback about a user's actions.
- *Simple error handling.*
- *User control*; user control over the interface should be nearly absolute. Again, this rule is violated by many IUIs.
- *Forgiveness*; actions should generally be reversible through an undo capability.
- *Reduce short-term memory load.*
- *Speak the user's language.*
- *Continuous representation*; objects and actions of interest should be continuously visible.

Some DM guidelines, such as consistency and user control are violated by IUIs. This is also the reason that many DM-interface designers heavily criticized some IUIs concepts (this will be further discussed in section 2.5). On the other hand, other guidelines are better served by IUIs than by DM interfaces. For example, using natural language IUIs can "speak the user's language" much better than DM systems. Also, many IUIs try to reduce the short-term memory load of users by taking over tasks.
The result of the development and implementation process is a user interface that has a "look-and-feel" that the designer thinks will suit the users and fulfill the requirements of the analysis phase.

2.3.3 Evaluation

In the evaluation stage of the design process we return to the questions of the analysis phase. The requirements that were drawn up in the analysis phase should be met and the effectiveness of the prototype system has to be investigated. To determine this effectiveness, usability measures should be specified. These measures may include the number of errors, task completion time, the user's attitude towards the interface, etc. A very important but subjective usability criteria is user satisfaction. Since the user needs to work with the interface he has the say about whether it is a good design and is pleasant to work with.

2.3.4 Refinement and tools

Based on the problems encountered in the evaluation stage, a number of design improvements will be made to the current prototype. Then, a new round of design, implementation, and evaluation is started. This iterative process will run until the result is satisfactory.
If proven successful the final interface technique of metaphor can be incorporated into existing user interface design tools.

2.4 Other design methods

The design process described in section 2.3 is often used in designing (commercial) software. However, many IUIs are research prototypes and do not follow these steps. Very often researchers/designers create an interface just to try out a new technique, neglecting (user) analysis and evaluation. Another reason that researchers do not follow an iterative design process is the lack of useful techniques for usability measurement, design tools and other useful design methods specifically applicable to IUIs.

2.4.1 Design guidelines

Many design guidelines for user interfaces have been developed, but only a few focus on intelligent interfaces. Several researchers have started to draw up some design guidelines specifically for IUIs. Below we list some of the suggestions found in literature [Benyon 1993], [Lieberman 1997], [Birnbaum et al 1997]:

- An adaptive user interface must be *developed in parallel with the application*. This is necessary since the designer continuously needs to focus on the system parts that need to be adapted.
- *Do not disturb the user's interaction*. It should always be possible for the user to ignore the proactive actions of the IUI. Suggest rather than act.
- *Operate in real-time*. Much of the benefit of an IUI comes from acting while the user is busy working with the system.
- *Take advantage of the user's think time*. When the user is thinking about what input to provide next, the IUI can take advantage of the available processing time, so it does not risk slowing down the user's interaction with the system.
- *Watch what the user is doing*. Take advantage of "free" information implicit in user actions.
- *Allow the user to choose his personal interaction style*. Different users like different interface styles and some techniques may be distracting or confusing to some users.

2.4.2 The general architecture

A big help in the design process can be the use of a general architecture. Several attempts to draw up a useful architecture have been made. For example [Benyon 1993] describes an overall architecture focused on user modeling and Mark Maybury and Wolgang Wahlster [Maybury and Wahlster 1998, p.3] present an architecture dealing with multimodal input. The picture below shows our interpretation of a general IUI architecture.

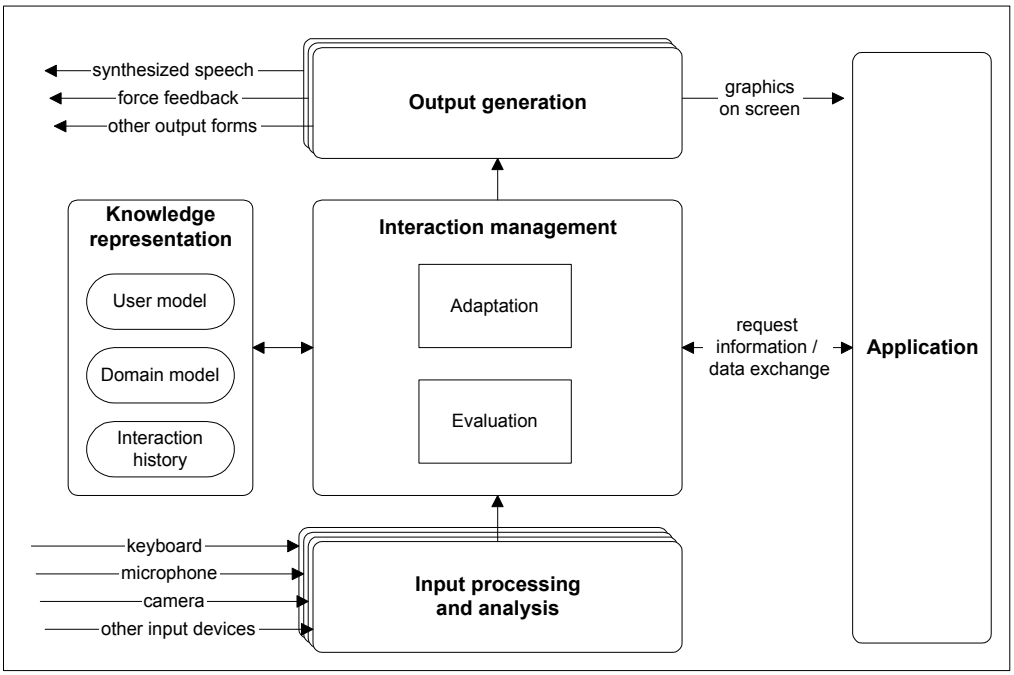

Figure 3: general IUI architecture

Input coming from the keyboard, mouse, microphone, camera, or possibly some other input device is recorded and then (pre-)processed. Processing includes labeling of events and other interesting input features. After each input modality has been analyzed, the separate modalities are fused together and evaluated. Note that in some cases it is desirable to do the fusion of input streams before the input processing, depending on the application and the features that need to be detected.
Once we know what input is coming in, we can start to determine the necessary course of action. First we have to evaluate what to do in the current situation. If there is information missing or if the user has requested information (e.g. the recorded speech contained a question from the user) this information will be requested from the application or some other external source. Usually there is an inference mechanism that draws up conclusions and updates the system's information: the user model, his interaction history, and information about the application domain. Once, all the necessary information is available and updated, the system must decide the best alternative for action. In the figure above we have called this adaptation, since usually some form of adaptation of the interface is chosen. Often, evaluation and adaptation occurs simultaneously using one inference engine for both, making the distinction between the evaluation and adaptation process is not quite clear. The chosen action still has to be generated, which is being done in the output generation part.
Most IUIs can be created with or fitted in this architecture, although often not all parts need to be explicitly modeled.

2.5 Criticism, problems and proposed solutions

The field of IUI is by no means mature and many basic problems still have to be resolved. This is probably one of the reasons that IUIs (and artificial intelligence for that matter) have received lot of criticism. The main problem is that many IUIs violate the well-accepted usability principles developed for DM systems. The point of concern usually is related to the adaptivity embedded into an IUI. Below we sum up several problems posed by some critics [Shneiderman 1997], [Shneiderman and Maes 1997],[Lanier 1995]:

- An adaptive system is *unpredictable and less transparent* than a DM interface. If a system can adapt its response and does not give the same response twice given the same input, then the system becomes unpredictable. This will hinder the user's comprehension of the system, making it impossible for him to do a successful action twice.

- Users are not in *control* anymore. An IUI can make decisions for the user, thus placing the user outside the control loop.
- IUIs often make *mistakes*. Many IUI systems use trail and error to determine a user's intent or preferences. Therefore users need to give feedback to the system or even resolve the mistakes made by the system.
- Simulated intelligence and adaptivity increases the risk of the user thinking that computer can do things that it cannot, thus creating *false expectations*. Especially with anthropomorphic agents (see also section 5.2) users may believe they can interact with the IUI just as with another person.
- Who is *responsible*? If a system can make decisions on its own, who is responsible for the actions: the programmer of the system, the user, or the system itself?
- What about the *privacy and thrust of the user?* What happens to the user profile that is created and maintained by an IUI? Can you guarantee that it is safe and will not be misused?

In part, the skepticism about IUIs originates from experiences with artificial intelligence techniques. In the early days of AI, many promises were made that later turned out to be unattainable. Since IUIs draw heavily on AI techniques some people are afraid the promises made by IUI researchers will go the same way. Another problem is that many IUI applications are designed just to show some neat technique or trick. Frequently, the focus of the designers is on the correctness or completeness of the program and usability and/or content is often forgotten.
The problems discussed above, form the main research issues in IUIs, so no ideal solutions have been found. Nevertheless, IUI researchers have proposed several possible solutions that will be discussed in the next sections.

2.5.1 Transparency and predictability

Less transparency is not necessarily a bad thing. One does not need to understand a system completely before you can work with it and abstraction can be very valuable to reduce cognitive load. As pointed out by Pattie Maes [Shneiderman and Maes 1997], we do not need to understand how a car works in order to drive it. The same goes for computer systems, both hardware and software for that matter. We do not want to know exactly how everything works before we can use it. Nevertheless, we do need to look carefully what functionality to hide from the user and what not. Kristina Höök [2000] has done research on the metaphor of a "black box" in a "glass box". In a system called PUSH, she hid the inference process of users' goals (black box) and showed a quite simplified view of what is going on to the user (glass box). This way, the user is presented with a relation between the inferred goal and the choice of adaptation, providing some means of predictability. What the user does not see is how exactly his goals are inferred from his actions recorded by the interface.

2.5.2 Control

A possible solution to the control problem is to give the users control over the adaptation process in an IUI. For example, Cook and Kay [1994] proposed that users should be allowed to inspect and alter the user model created by the system. Unfortunately, this introduces new problems. The user usually does not understand the effects of altering adaptivity parameters. In addition, we have given the user an additional task to perform, distraction him from his original tasks. Basically this solution only shifts the problem and will only be feasible in rare cases where the user knows exactly what he or she wants.
A better solution is to make the adaptivity part of the interface and system design and make sure that there are means for the user to directly or indirectly correct the system's choices for adaptation. Again allowing the user to accept, refuse, or ignore the proposed adaptation is a good solution.

2.5.3 Mistakes

The amount of mistakes current IUIs make is still a problem. Most mistakes are caused by lack of knowledge, and the uncertainties that are inherent in the input. Expectations are that with more

information from different input sources and new reasoning techniques to process them, the number of errors will be decreased dramatically.

2.5.4 False expectations

People have a tendency to attribute intelligence to the anthropomorphic creatures that are found in some IUIs. These agents try to help people performing a certain task. Their human-like behavior however, raises the users' expectations regarding the agents intellect and capabilities.
For the credibility of IUIs, it is important that interfaces are designed in such a way that they create the right expectations in the user: neither too high nor too low. However, not much is known about which cues in an interface give rise to which expectations in the user and this subject of studied further.

2.5.5 Responsibility

The false and high expectations of people can also give rise to another problem. If people regard an interface agent as a responsible entity or even as a fellow human, then the users' feeling of responsibility will diminish. It was the agent who deleted the files, not me! As far as we known, there is no law or precedent that handles that issue of agent responsibility, so the question whether intelligent agents (or otherwise their user or designer) can be held responsible for their actions is still open. Again, further research should point out how the user's expectations and feeling of responsibility can be set straight.

2.5.6 Privacy and trust

All IUIs that contain information about the user, force the user to accept that the systems contains information about them. Whether this is acceptable will depend on the nature and sensitivity of the information, as well as how much the users tolerate and trust the system. Cook and Kay's proposal to allow users to investigate the user information stored in an IUI [Cook and Kay 1994] can help in this case. A user's trust becomes important when the IUI takes over tasks from the user. How much is the user willing to let go? Can the user trust information from other people's IUI? Do you want the agent to notify you when it adds some user information or automates some task, and if so when? If the IUI makes just one mistake, the trust of the user in the system will immediately be diminished, so IUI designers should be very careful about this, providing clearity about what happens to their personal information.

The problems discussed above and the fact that IUIs still have to prove their selves are responsible for the very slow introduction of IUI in commercial products. However, this is now slowly changing. Simple IUI-features are starting to appear in more and more products, usually as an extra feature. Expectations are that first these simple IUI-features will become more intelligent and complex. Once the public is accustomed to them, these extra features will become larger parts of the interface.

Chapter 3: Communicating with the user

Traditional computer interfaces consists of a monitor, a mouse and a keyboard. IUIs incorporate a much broader range of input and output devices. This chapter describes a number of communication methods that are typically used in IUIs. After a short introduction in section 3.1, we start with one of the most popular new communication methods, which are natural language interfaces (section 3.2). Then we will look at the recognition of human gestures in section 3.3. Section 3.4 deals with image recognition methods that give more information about the user of an IUI,, such as gaze tracking, facial recognition and lip reading. The chapter closes with our investigation of the combination of several input and output methods (multimodal interfaces) in section 3.5.

3.1 Introduction

Human-human interaction is quite different from human-machine interaction. Normally, human-human communication goes in an incremental way, through a dialogue. The dialogue consists of presenting information (talking) and assessing whether the other person understands the information that is conveyed. Normally in a human-human dialogue there are pauses to allow the other person to provide feedback. This way, both persons can influence and steer the conversation.
Computers on the other hand rarely use such a dialogue. Computer systems usually expect little feedback other than pressing a button and they do not look if the user has understood the information. For example, asking the same question twice to a help system usually results in giving exactly the same answer twice, instead of providing a different, perhaps more clear explanation. The main reason that human-computer interaction is much more rigid is because computers cannot deal with all the additional information that people usually convey when communicating. For example, a computer interface cannot recognize a surprised face or pointing gesture to an object (other than with a mouse). Research on IUI-communication methods is trying to create interfaces that can deal with the more subtle communication methods used by humans.

3.2 Natural language systems

A natural language system uses speech or written text to communicate with a user. The natural language system tries to recognize and understand the utterances or typed messages of the user and returns a relevant answer. Although we are not yet capable of creating such a system (at least not one that is as accurate and knowledgeable as a real person) a lot of research has been done that can make this possible. Any good natural language system would require at least the following components [Wyard et al 1996]:

1. *Speech recognition*; conversion of an input speech utterance into a string of words.
2. *Language understanding*; analysis of the string of words (as much as possible) to extract a meaning representation for the recognized utterance.
3. *Dialogue management*; controlling the interaction or dialogue between the system and the user, which includes coordination of other components of the system.
4. *Database query*; retrieving the information requested by the user.
5. *Response generation*; specification of the text that is to be the output message of the system.
6. *Speech output*; actual generation of the output message using text-to-speech synthesis or pre-recorded sentences.

Note that for natural language systems dealing with written text a subset of these points (at least 2 – 5) applies. We will discuss the points further in the next sections.

3.2.1 Speech recognition

Speech recognition is the process of transforming a continuous speech signal into a form that can be understood by a computer (text). For this, the computer first has to detect when someone is speaking. Then the detected speech is separated into fragments, either words or phonemes, and these fragments are then analyzed. Using statistical methods the probability of possible words, sentences or phonemes is calculated and the one(s) with the highest probability are the result of the recognition process.

Although several methods for speech recognition have been devised, the most successful is the Hidden Markov Model (HMM). A HMM encodes how likely a certain utterance (phoneme or word) is, given the previous phonemes or words. At a higher level, a language model is used to contain information about which words or string of words are more likely than others, given a point in the dialogue. Speech recognition with HMMs and language models still is not perfect, but can be done reasonably fast and reliable. Further improvements in speech recognition are expected to be made mainly through greater computing power than on major theoretical advances.

Speech recognition is one of the most popular research topics in IUIs and has received a lot of attention in the last 25 years. Recently, several good commercial speech-recognition systems. designed by big companies such as Philips and IBM, have appeared on the market. Speech systems are now used for dictation on personal computers (and recently also on small palm computers), for learning foreign languages, and for automating telephony applications. The main advantage of speech-recognition systems is the ease of use (no learning is involved), the relatively high speed of interaction, and the fact that the user's hands are free for other tasks. A problem in speech recognition applications is the limited vocabulary of the recognizer. All words that need to be recognized need to be stored in the system and the recognizer will fail on words outside its vocabulary. Also, the more words are stored, the more chance the recognition process has to make mistakes and the slower the system becomes. A possible solution is to restrict the search area during the recognition process based on the already recognized words.

More on speech recognition can be found in [Rabiner and Juang 1993] and in [Jelinek 1999].

3.2.2 Natural language understanding

The next step in a natural human-computer dialogue system would be language understanding. A language understanding system tries to give meaning to a sentence or a group of words. This sentence or group of words consists of spoken words recognized by a speech recognition system or simply a text that is typed by the user with a keyboard. Together with speech recognition, language understanding is also named natural language processing.

The meaning of a sentence is derived by the meaning of the parts that make up the sentence. We can try to distinguish the type of utterance that the user made, the expectations present in the utterance, or the entities that are referred to and the relationship between them. In theory, we can use a combination of syntactic and semantic analysis (parsing) to determine these properties. In practice, this will take too much time and is much too knowledge intensive. Practical language-understanding systems use keyword or phrase spotting to find the main concepts and then only in a limited task domain.

A problem in language understanding is that the output from the speech recognition process often is not in the form of a grammatically correct sentence. Human speech includes fragments, self-corrections, and grammatically incorrect utterances. In addition, the speech recognizer might be incorrect in determining the highest probability of one or more words, thus providing the language-understanding component with a wrong sentence. A better approach would be to analyze all likely combinations from the speech recognizer and afterwards let the language-understanding component decide which combination is most likely correct. However, this will increase the load on the language-understanding component. A third problem is that of ambiguity. Ambiguity arises when the meaning of a particular word can only be deduced from the context. Anaphoric references and ellipses (something that is left out but can but can be determined from previous utterances) are still very difficult for a natural language system to deal with. Fourth, there is the problem of creating a suitable representation of all the deduced meanings for future reference. This representation must

be exact (unambiguous) and complete (contain all relevant possible meanings for the application). Usually a logic form is used for meaning representation.

3.2.3 Dialogue systems

A dialogue system, also called dialogue manager, is the most central part of a natural language system. It manages the conversation and coordinates calls to other system components. Strictly speaking, the job of the dialogue system is simply to engage into a dialogue with the user. However, just asking the user to respond to the computer's questions with yes or no can also be classified as a dialogue. Therefore, we require dialogue systems to be as flexible as possible, using a natural style of interaction.

As with language understanding, the ideal dialogue system processes human speech, but very often, for simplicity reasons, dialogue systems only deal with typed sentences. Since it is very hard to create a dialogue system that can really pass of as a human being with common knowledge on many topics (remember the Turing test [Turing 1950]), most dialogue systems have fairly small vocabularies, restricted to a small problem domain. Some artificial characters on the web try to respond to more than one topic (for example [Extempo 2002]) but still these characters have limited understanding and often make strange remarks when they do not understand you. Especially anaphoric references (using information provided earlier in the conversation) are very difficult.

Application 1; the Alparon dialogue manager

As discussed earlier, in order to further reduce complexity many dialogue systems try to control the dialogue as much as possible. An example of a flexible dialogue system is the Alparon dialogue manager developed by Robert van Vark at Delft University of Technology [Van Vark 1997]. The Alparon dialogue manager is part of the Personal Intelligent Travel Assistant (PITA) project. The goal of the PITA project is to provide a user with information about public transport, before and during a trip (e.g. time of departure, transfers, delays etc). The Alparon dialogue manager uses multiple modules to isolate dialogue phenomena such as disambiguation, context updating, response generation (see also Figure 4). The coding scheme used by the Alparon dialogue manager was constructed after analyzing a corpus of over 5000 human-human dialogues recorded at a public transport-information call-center. Before a trip is taken, the Alparon dialogue manager engages in a conversation with the user in order to find out the user's wishes. The dialogue manager searches every comment made by the user for useful information, such as the destination of the user, the time of departure etc. Then it looks for the most important missing piece of information that is necessary to help the user and will ask the user to provide this. Once all the necessary information is provided the system can give the appropriate information.

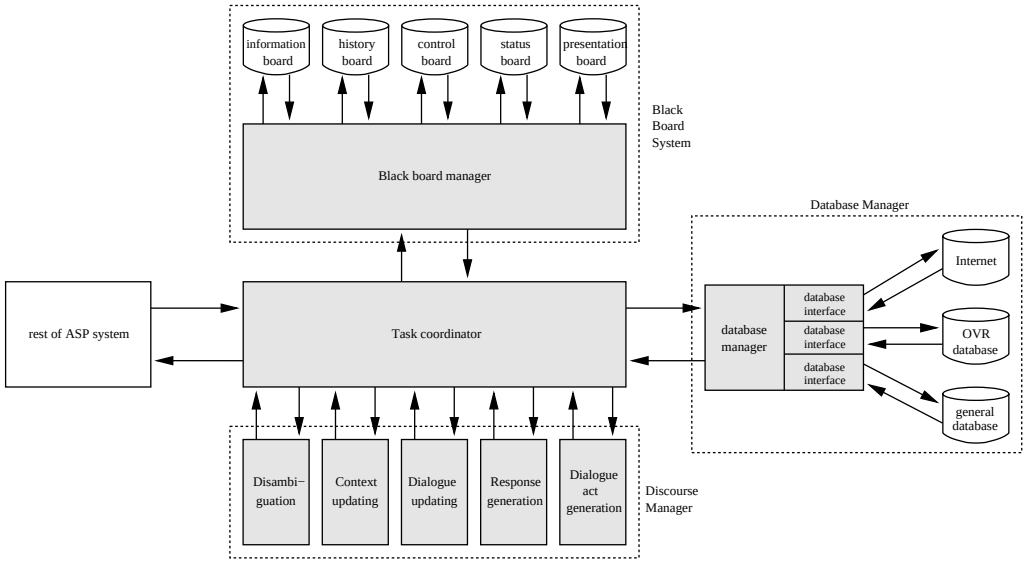

Figure 4: Architecture of the Alparon dialogue manager, taken from [Van Vark 199]

Application 2: mixed-initiative dialogue systems

A special type of dialogue system is the mixed-initiative dialogue system. In a mixed-intiative dialogue system the system is running alongside a user program. Both the user and the system can start a dialogue at any time.

An example of a mixed-initiative dialog system is the LookOut program designed by Microsoft Research [Horvitz 1999]. Recognizing typical patterns of expressions about meetings, LookOut tries to detect appointments in email messages and schedule these appointments in the user's electronic agenda in Microsoft's Outlook program. The user can interact with the system via direct manipulation or by talking to an anthropomorphic interface agent. The system can be set to manual mode where the user takes the initiative to schedule appointments, or to an automatic mode, where the system comes with a proposal for a new appointment. LookOut will always provide feedback to the user to make corrections or refinements.

3.2.4 Database query

Natural language systems can provide an interface to a database or some other data set, e.g. stored in an application. It is common for a user to request information from the available data. When the dialogue manager has identified that the user has made such a request, or that it needs data before it can continue the dialogue, the dialogue manager must send a query to a database. This query either is successful or unsuccessful. When it is unsuccessful the database query component must pass as much information as possible back to the dialogue manager about why it has failed. Then it is up to the dialogue manager to decide what to do next.

3.2.5 Speech output

An often-used approach for speech output is to use pre-recorded words or sentences and play them back at the appropriate time. Although this provides high quality output, these systems are not generic and require all possible output to be specified and recorded in advance. In addition, much storage space is needed to store the used sentences in a large text output system. A better approach is to parse the text and create speech output based on phonemes. Any reasonable phoneme text-to-speech output system can cover an entire language or sometimes even multiple languages. However, the quality of the output is still quite poor, with metallic voices and strange intonations.

3.2.6 An example

Many natural language systems do not contain all the components described in the sections above. Most systems do not use speech recognition, and often the speech output part is not present. Instead, these dialogue systems use plain text as input and output. One of the few (prototype) systems that do use speech recognition, natural language understanding, dialogue management and text-to-speech output is the Persona project done by the Microsoft research [Ball et al 1996]. The goal of the Persona project was to study natural interaction between user and computer. The idea was to create one user interface containing all components discussed above to allow spoken conversation with a lifelike computer character (anthropomorphized agent). The resulting prototype system, which is shown in Figure 5, features a parrot called Speedy that selects and plays songs from stored music albums or gives information about the available albums as requested by the user. Using pre-recorded sentences and real-time video output rendering, Speedy can talk back to the user's requests and behave realistically. Although response latency is several seconds and a special graphics machine had to be used for graphics rendering, the resulting interaction (animation) is quite good.

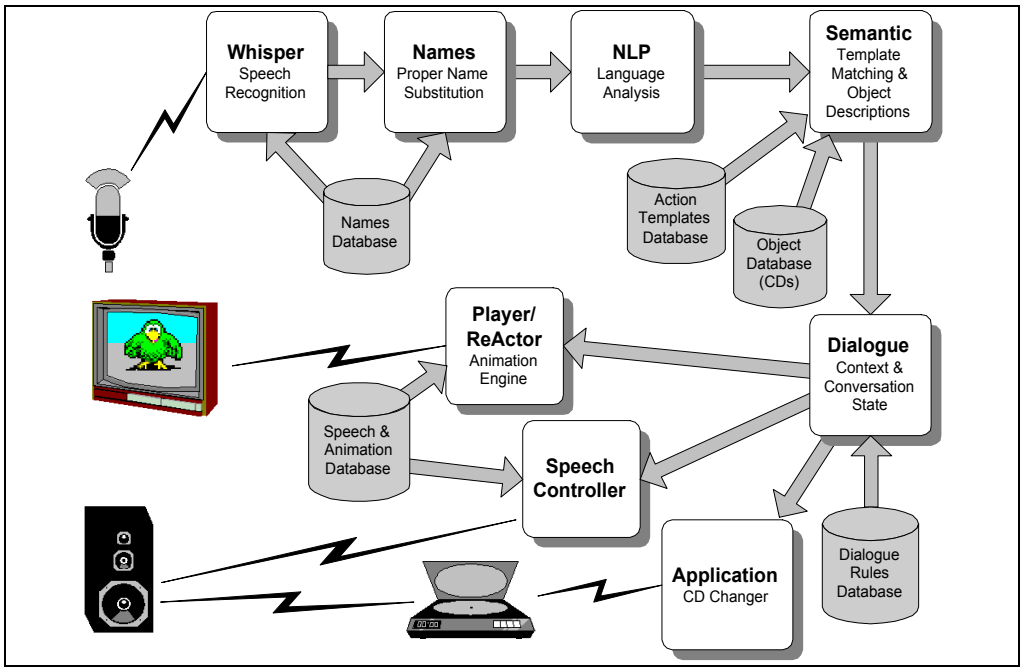

Figure 5: System diagram of the Persona conversational assistant, taken from [Ball et al 1996]

3.3 Gestures

People use gestures a lot when talking, often subconsciously. The most often-used gestures are those associated with speech, for example pointing to an object that one is referring to. This kind of gesturing is called **gesticulation**. Gestures that function independent of speech are called autonomous gestures, for example sign language.

The goal of gesture recognition research is let a computer system recognize and understand specific human movements that are made to convey information. There are several ways of making a gesture but the most important methods for HCI are hand gestures (e.g. pointing), pen- or mouse-created gestures (e.g. handwriting), and human-body motion gestures (e.g. nodding yes). Popular gesture input methods are touch-screens, mouses, computer vision (image differencing), electromagnetic fields (field distortions, and datagloves). Several taxonomies have been devised that categorize the different types of gestures. An often-cited taxonomy is that of Rime and Schiaratura [1991]:

- *Symbolic gestures* have a (single) verbal and often cultural dependant meaning, for example the OK sign, or sign language for deaf people.
- *Deictic gestures* are made by pointing or motioning to direct attention to some object or event.
- *Iconic gestures* are gestures that display information about the size, shape or orientation of objects, spatial relations, and actions, for example using hands to indicate the size of fish that one caught).
- *Pantomimic gestures* consist of manipulating an invisible imaginary object or tool, for example making a fist and moving to indicate a hammer).

Symbolic gestures can be identified most easily by a gesture recognition system. Deictic, iconic, and pantomimic gestures usually require additional information (context) and thus are harder to recognize. A problem with gesture recognition, and especially that of 3D gestures, is how to find the segments that are important. A simple method is to select segments based on movement. For example, when the motion of a person's hand stops the end of a segment is reached.

A very popular topic in gesture recognition research is that of 2D handwriting recognition. The idea is to recognize a person's writings while he or she is writing it, not to be confused with optical

character recognition on an already written text. Although recognition is far from perfect, there are now hand-held devices on the market that are capable of handwriting recognition. It is reported that these devices can reach a recognition rate of 95%, although in every day practice it is very hard to achieve this rate. Also, these devices require the use of slightly altered and simplified characters in order to successfully identify the hard-to-recognize characters.

3.4 Image recognition

Image recognition or computer vision can be used for many purposes, ranging from facial expressions recognition to robot navigation. Image recognition in IUIs is mostly used to obtain specific information about the user of the system.

3.4.1 Recognizing faces and facial expressions

Research has shown that it is possible for a machine to automatically recognize facial expressions. Using a camera to obtain images, automatic recognition of facial expression is done using the following steps:

1. Find the face in the image
2. Detect facial features
3. Classifying the observed features

At the Data and Knowledge Engineering group at Delft University of Technology the ISFER (Integrated System for Facial Recognition) system was developed that is able to automatically classify the emotions of a person [Pantic 2001]. ISFER uses the Facial Action Coding System (FACS) developed in 1978 by Paul Ekman and Wallace Friesen [Ekman and Friesen 1978]. FACS uses action units (specific points in a person's face) and the movements of those points to determine the emotion. For example, if a person's is angry he will squint, moving the action units placed at the top and bottom eyelids, close together. Using examples that were given in a training session, ISFER will rate the intensity of each emotion, making "mixed emotions" possible (e.g. (e.g. 70% happiness, 20% nervous).

3.4.2 Gaze tracking

People use their eyes with very little conscious effort. Most of the time, we automatically look at the object we are working on. To see an object clearly, it is necessary to move your eyes so that the object appears on the fovea, a small area at the center of the retina. The fovea covers approximately one degree of visual arc. Because of this, a person's eye position provides a rather good indication (to within the one-degree width of the fovea) of a person's focus point of attention on a display [Jacob 1991].

A gaze tracker is a device that is mounted either on a person's head or is placed remotely in front of the person (e.g. on the desktop). A gaze tracker sends out infra-red light and captures reflections of this light from both the corneas and the retina of a person's eye. Input from a gaze tracker consists of a continuous stream of raw data that usually specifies the x and y coordinate of a position of the point of gaze. Generally, eye position is recorded every 1/60 of a second or more. The recorded data is first filtered for noise and sometimes the data is compensated for head movements (when this is measured). Then the meaningful events are recognized. Most likely this will be saccades (sudden eye movements) and/or fixations (200-600ms periods of relatively stable eye position). Besides fixations, other measurements that can be taken with a gaze tracker include; the point of gaze (e.g. a specific object), fixation duration, scan pattern (and its randomness), pupil diameter and blink rate. Some of these measures have been used to determine the workload of pilots [Svensson et al 1997] and air traffic controllers [Brookings et al 1996].

A big advantage of gaze tracking is that it is not very demanding to the user. As the user is doing his work, the system can track his gaze without the user being aware of it, provided that a desktop-mounted gaze tracker is used. Head-mounted eye trackers are slightly more intrusive, but can follow a person's gaze in all directions.

Application

Just like gestures, eye movements can be interpreted in several ways. In all cases, context plays an important role. Rob Jacob [1991] has explored interaction techniques using natural eye movement. In his experiments he measured the visual line of sight by simultaneously tracking corneal reflection and pupil outline using a light shown at the eye. A servo-controlled mirror compensated for user head motion. Using a fixation-recognition algorithm Jacob developed several gaze interaction techniques for object selection, attribute display and object movement. Objects are selected by looking at an object for a certain period (dwell time) or by looking at an object and pressing a button on the keyboard. Using a dwell time between 150-250ms proved to give good results for "normal" objects. For menu selection a longer dwell time of 400-1000 ms was needed since the user needs time to read the available menu options. Note that there is no standard technique for determining fixations and that a minor change in the determination method can lead to very different results. The different methods and different gaze tracking equipment have led to different definitions of fixations in the literature, which makes a comparison of results very difficult.

3.4.3 Lip reading

A problem with speech recognition based on an acoustic signal, which was discussed in section 3.2.1, is that it functions very badly when there is a lot of noise. The reason for this is that it is very hard to distinguish the speaker's voice from other sounds. With lip reading the idea is that the computer tries to determine what someone is saying just by looking at the person's lip movements, like a deaf person. By analyzing video images of the mouth and using geometric features, such as the width and height of the lips, the most probable sound (phoneme) can be determined.
When there is little noise lip reading is much less accurate than acoustic speech recognition, but its advantage is that performance does not degrade with more noise. Especially in noisy environments lip reading can play in important role in reducing the error rate of a speech recognition process, so combining both recognition methods can be valuable. The figure below shows a possible method for combining lip reading with acoustic speech recognition.

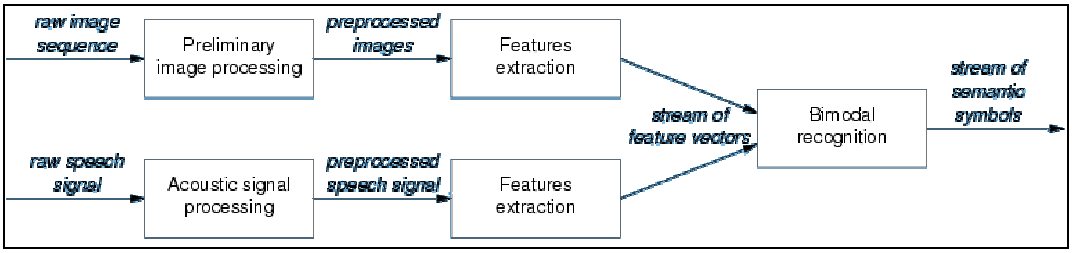

Figure 6: Bimodal speech recognition using lip reading and acoustic processing

A problem that arises when combining lip reading and acoustic recognition is that audio and video signals are often not synchronous. For example, when you pronounce the letter "p", you first press your lips together (which is seen on the video images) and a short time latter the actual sound is created by releasing the lips. Therefore the two channels need to be time stamped, or otherwise be synchronized.

Example

An example of a lip reading system is the prototype system created by Jacek Wojdel [Wojdel and Rothkrantz 2001a]. Wojdel's system starts by using a red-sensitive filter to detect the mouth. The resulting image contains the red lips of the mouth, but also some noise produced by other red pixels of other parts of the face. Therefore a statistical probability distribution calculates the geometrical features of the mouth and determines its location. Then an Elman neural network is applied to the found geometrical shapes to discriminate between vowels and consonants. The resulting system is fairly robust (not sensitive to image noise) and person independent. In an experiment determining

what numbers are pronounced by a person, Wojdel's system achieved a success rate of 75-91% [Wojdel and Rothkrantz 2001]. In comparison, deaf persons achieve about 65%.

3.5 Multimodal interfaces

When people communicate they use multiple ways to convey information. You might expect that speech is the most important method of communication between people, but research has shown that body language (gestures and facial features) are just as important.
With multimodal interfaces, the idea is to use multiple input channels in human-computer interaction. Instead of only speech or text as input, the system can use image recognition to look at a user's face or gestures. This way, information from one mode of interaction can complement the information received from another mode. Multimodal interfaces try to integrate speech, written text, body language, gestures, eye or lip movements and other forms of communication in order to better understand the user and to communicate more effectively. Of course, the choice of the used modalities in multimodal interfaces depends very much on the application of the system.
When users can choose from multiple modalities to interact with a system, the system has the potential to be accessible to a broader range of users, for example people with disabilities. Also, multimodal user interfaces are much more robust than normal interfaces, at least in theory. Processing input from one modality can be simplified by using information from another and the user can choose the modality that is the least error prone given the circumstances.

3.5.1 Data fusion

The main bottleneck in multimodal interfaces is to combine the information gathered from the used modalities in real time. All multimodal interfaces need some form of input coordination or fusion system to synchronize related input.
We can distinguish between systems that integrate input signals very early, during the (pre-) processing of the input signals, and between systems that integrate after processing each of the input signals. The first method is called **feature-level fusion**, whereas the latter is called **semantic-level fusion**. Feature-level fusion is appropriate when combining two related modalities, such as speech processing and lip movements. The advantage of feature-level fusion is the improved recognition rate due to the complementary nature of both input channels (mutual disambiguation). A drawback of this tight integration of input processing is that the system has to be "retrained" when the one of the input modalities is changed. In addition, training a system with multiple modalities using simultaneous input is much more difficult than training one modality at the time. Semantic-level fusion is much easier. It does not have the benefit of direct complementary information, but a semantic-level fusion system is much easier to create and extend. Such as system can use multiple off-the-shelf recognition techniques and new modalities can easily be added later. Semantic-level integration is usually done either through unification of existing data or by looking for missing data. The latter is called **frame-based integration**, where the systems tries to fill missing data slots.
All multimodal (semantic-level) input systems need some kind of time stamping in order to combine and interpret a combination of input measures. Time stamping should occur at least at the beginning and end of each input signal. Unlike traditional DM interfaces, multimodal input is usually not unambiguous and requires probabilistic processing methods.

3.5.2 Applications

Almost all multimodal interfaces are still in the research stage and commercial applications are scarce. The most often used modality in multimodal interfaces is speech recognition. This is probably because speech is the most natural and direct method for people to communicate and speech is very much suited to be complemented by other forms of interaction.

Natural language and direct manipulation

Direct manipulation and natural language input are very much complementary to each other. With natural language the user can simply tell the computer what action to perform on what object. However, speech can be, and often is, ambiguous and context-sensitive. Also, using only natural

language, a user cannot see what actions are available and which spoken commands the computer will understand. On the other hand, DM shows the user what the possible actions are and it reduces ambiguity. Unfortunately, DM has limited expressiveness and the user first has to navigate through menus to select objects or actions. Using a combination of natural language and direct manipulation, we can benefit from the strong points of both methods.
An example of a multimodal system using natural language and direct manipulation is built as part of the CUBRICON project [Neal et al 1997]. The CUBRICON system accepts both spoken and typed natural language sentences in combination with pointing with a mouse. The system generates output on a display and through synthesized speech. In addition, the system can also make "gestures" and draw the user's attention to an object on the screen by letting the object blink or by drawing a circle around it.

Natural language and gestures

Another possibility is to combine natural language input with gestures. It is well known that people often use gestures when talking, so it seems only natural to incorporate this into the speech recognition process
One of the first people to combine natural language and gestures is Richard Bolt [1980]. In his groundbreaking paper about the "Put-that-there" interface, Bolt describes a room with a wall-sized screen on which some objects are displayed. The user, placed in a chair with two joysticks and a touch screen, can manipulate the displayed objects either by using the joysticks and touch screen (direct manipulation) or by making gestures and talking to the computer at the same time. Gestures consist of pointing to the screen with your hand. The user can use pronouns, such as "that" or "there", while pointing to indicate a certain object or location.

Natural language, gestures and gaze tracking

A drawback of deictic gestures is that pointing is inexact. Referred objects may overlap, or the user may inadvertently point just next to the intended object. David Koons, Carlton Sparrell and Kristinn Thórisson tried to solve this problem by using gaze tracking. In their paper they describe two prototype applications that use a combination of input from speech, deictic gaze (looking at objects) and hand gestures to resolve references to objects in the system [Koons et al 1997]. Their frame-based and time-stamped interpretation of input allows creating a hierarchy of increasingly complex input evaluation. Whenever information from an input channel (e.g. speech) is missing, the input from other channels (e.g. gestures) is used to fill in the blanks as much as possible. This way uttered sentences such as "Move this object" can be understood.

Chapter 4: Learning from the user

In this chapter we will discuss how an IUI, or a machine in general, can learn from its interactions with the user. In the first three sections we will provide some background knowledge about artificial intelligence techniques used for learning. We will discuss learning definitions, knowledge representation methods and machine learning techniques (section 4.1, 4.2 and 4.3 respectively). In section 4.4 we will look at user presentations derived from interactions with interfaces, which is called user modeling. Then, in section 4.5 we will show how plan recognition is very much useful for user modeling.

4.1 What is learning?

Just like intelligence (see the discussion in section 2.1), the concept of learning is difficult to define. Many different definitions of learning exist. In Merriam-Webster's online dictionary [Merriam-Webster 2003] learning is defined as "*to gain knowledge or understanding of or skill in by study, instruction, or experience*". Webster's definition fits the general idea of learning very well, but when we look at computer systems a more accurate definition is given by Meystel and Albus [2002]:

> *"Learning is a process based on the experience of intelligent system functioning (their sensory perception, world representation, behavior generation, value judgement, communication etc.) which provides higher efficiency which is considered to be a subset of the (externally given) assignment for the intelligent system".*

The basic idea behind learning in any machine is to deal with unforeseen situations and unknown circumstances. An IUI can use sensors to measure and draw up a world representation. Based on this representation a certain action or behavior can be generated and communicated to the user. We can distinguish three types of knowledge that would be useful for an IUI to acquire:

1. *Unknown information*: the information necessary for the IUI is simply not available because it is not known beforehand, for example the preferences of an individual user.
2. *Dynamic information*: even if we have a complete model of the user and its environment to begin with, this knowledge could quickly become obsolete due to changing preferences or environments.
3. *Hard to program knowledge*: information that is very difficult to program by hand may be learned by showing examples.

Through experience the IUI may discover new methods to help the user. Automated learning in IUIs is especially useful if you do not want to bother the user asking information about what he wants. For example, instead of asking a user which tasks it wants the IUI to perform, the interface can watch what the user is doing and propose to automate a task that it has seen the user perform often. Another situation where automated learning is useful is when the user does not know how to instruct the IUI or when the user is not able to express his own preferences.
Roughly speaking, we can distinguish two broad classes of IUI applications that use learning; informative interfaces and generative interfaces. The first deals with selection and filtering of information, for example email-filtering or recommender systems. Generative interfaces try to generate useful new knowledge structures to satisfy the user's needs.

4.2 Knowledge representation and reasoning

Learning implies that you "know" something. In other words, there must be some form of stored knowledge. When people learn, they store knowledge in the various parts of their brain (scientists are still trying to figure out how this works exactly), but since computers do not have a brain, they must use an alternative way. Knowledge can be stored in a computer memory by filling it with certain symbols; numbers and characters that represent a certain fact or belief. Each symbol

represents an object or idea and this is called **"knowledge representation"**. A knowledge representation language defines the syntax and semantics for expressing knowledge. The language should be capable of expressing all the necessary descriptions in an appropriate and effective way. The application domain determines the type of knowledge representation that should be used. One has to consider four important features [Tecuci 1998]:

1. *Representational adequacy:* the ability to represent all of the types of knowledge that are needed in a certain application.
2. *Inferential adequacy:* the ability to use the available knowledge to deduce new information (reasoning).
3. *Problem-solving efficiency:* the ability to represent and create the knowledge and procedures needed to solve the problem.
4. *Learning efficiency:* the ability to easily acquire and learn new information and integrate it within existing knowledge structures.

Very much related to the choice of the knowledge representation language is the reasoning system, also called inference system. A **reasoning system** consists of data structures for storing knowledge and, more importantly, procedures for manipulating these structures and deducing new facts. The choice of knowledge representation dictates the type of reasoning. Examples of often-used knowledge representations and reasoning systems are predicate calculus, semantic networks, frames, production systems, Bayesian belief networks, and fuzzy systems. Many other knowledge representations exist and the interested reader is referred to [Russell and Norvig 1995] for more detailed information.

4.2.1 Predicate calculus

Predicate calculus is a formal logic that uses predicates on objects to define attributes and relations between objects. Two techniques, called resolution and unification, are used to process predicate statements and check whether a particular statement is true or not. Resolution is an algorithm for proving that facts are true or false by showing that the negation of the fact is not true. Unification is a technique that takes two sentences in predicate logic and tries to match and replace their variables. Together these algorithms form the basis of the programming language Prolog.
Predicate calculus scores high on representational and inferential adequacy, but low on problem-solving efficiency. It is very difficult to implement learning methods due to the complexity of predicate calculus, but the integration of new knowledge is fairly easy because of the modularity of the representation.

4.2.2 Semantic networks

Semantic networks are graphs in which the nodes represent objects, situations or events, and arcs between the nodes represent the relation between them. Semantic networks are well suited for representing knowledge about objects, but less so for representing processes. Their inferential capacity is reasonably high, but their learning capacity is rather low due to the difficulty of adding new nodes to existing networks.

4.2.3 Frames

A frame is a collection of attributes that define the state of an object and its relationship to other frames (objects). A frame contains slots that represent attributes and fillers or scripts, which are attached procedures that are called when the value of a slot changes. Frames are often linked into a hierarchy to represent the hierarchical structure of objects, similar to semantic networks. In this case, frames at a certain level can inherit knowledge from higher-level frames.
Frames and semantic nets are closely related, both represent objects and their relations with other objects, and both have approximately the same advantages and disadvantages. The main differences between frames and semantic nets is the higher expressive power of frames and the syntax of the two representations.

4.2.4 Production systems

Production systems represent what to do in certain predetermined situations, but are less adequate for representing knowledge about objects. The IF-THEN rules used by production systems are very easy to understand and new knowledge can be added easily. This is the reason that production systems have become one of the most popular forms of declarative knowledge representations in AI. A typical production system consists of the following parts:

- *Knowledge base:* permanent memory containing IF-THEN rules.
- *Working memory:* temporary memory containing new or derived facts.
- *Inference engine or matching mechanism:* reasoning logic used to produce new data.
- *Conflict resolution mechanism:* procedure that decides which rules should be executed.

A production system works by matching the facts stored in the working memory to the IF-part of the rules stored in the knowledge base. The conflict resolution mechanism chooses the best rule(s) of all matching rules. The THEN-part of these rules are then executed.
An example of a production system is the "**C** **L**anguage **I**ntegrated **P**roduction **S**ystem" or CLIPS for short [CLIPS 2003].

4.2.5 Bayesian belief networks

Bayes' theorem says that the conditional probability that event Y will occur given that event X already occurred can be computed, providing we know the prior probabilities that X and Y could happen, and the conditional probability that X will occur when we know that Y has already occurred. In mathematics this is described as:

$$P(Y \mid X) = P(X \mid Y) * P(Y) / P(X) \qquad \textit{Equation 4-1}$$

A Bayesian belief network, also called causal network or probabilistic network, is a directed acyclic graph in which nodes stand for propositional variables and edges for the probabilistic relation between them.
A Bayesian belief network represents knowledge in its nodes and links and can modify information propagated among the nodes. The knowledge stored in the network can be specified a priori or learned from examples. The primary use of Bayesian networks is to use probability theory to reason with uncertainty.

4.2.6 Fuzzy systems

Fuzzy control systems produce actions using a set of fuzzy rules based on fuzzy logic. In conventional logic assertions about the world are either true or false, there is nothing in between. Values such as true and false (1 and 0) are referred to as crisp, that is, they have one exact meaning. Fuzzy logic allows variables to take on other values (between 0 and 1), determined by how much they belong to a particular set. In fuzzy logic these variables are referred to as linguistic variables, which have non-crisp meanings (e.g. fast, slow, far, big, etc.). Membership functions measure the degree of similarity an instance of a variable has in its associated fuzzy set. A fuzzy logic control system consists of the following:

- *Fuzzifier:* maps a set of crisp inputs to a collection of fuzzy input sets.
- *Fuzzy rule base:* contains a collection of IF-THEN rules.
- *Fuzzy inference engine:* maps fuzzy sets onto other fuzzy sets according to the rulebase and membership functions.
- *Defuzzifier:* maps fuzzy output sets onto a set of crisp output commands.

The main advantage of fuzzy systems is that they are more flexible than conventional rule-based methods and fuzzy rules are often better understandable by humans due to the use of linguistic variables.

4.3 Machine learning techniques

Most knowledge representations allow deriving facts from existing knowledge in a fairly straightforward and simple manner, for example by using IF-THEN rules. However, these representations and deduction rules are entered a priori and are activated only after a certain specified event occurs. For more sophisticated "learning" we need machine learning methods. **Machine learning**, a subfield of AI, tries to improve performance in some task domain based on partial experience with that domain. Machine learning implies induction (generalization) beyond the training data. Examples of machine learning applications in IUIs are demonstrational interfaces, user modeling and plan recognition, which we will discuss later in this chapter.

4.3.1 Reinforcement learning

Reinforcement learning is probably one of the most popular methods of machine learning. Reinforcement learning systems attempt to learn by exploring all possibilities in all of the available states (trail-and-error) and by ranking the possible actions in the order of appropriateness or usefulness. This appropriateness is determined by an evaluation mechanisms that sends the necessary reinforcement signal to the control system. When evaluation is performed by a human, we call it supervised learning. When evaluation is done automatically, for example by another program, we call it unsupervised learning. The feedback to the learning system contains information about the quality of action. It may be as simple as a binary pass/fail or a more complex numeric evaluation. There is no specification as to what the correct response would be, only how well the particular response worked.
The problem of learning an optimal strategy consists of searching for paths connecting the current state with the goal in the state space. The longer the distance between a state and the goal, the longer it takes to learn the strategy. Breaking the problem into smaller subproblems effectively shortens the distance between the reinforcement signal and the individual actions, but this requires built-in domain information.
One of the major problems of reinforcement learning is **credit assignment**. It is hard to determine which of the individual components is largely responsible for the success or failure of a response. Another important weakness of reinforcement learning is the unstructured use of the input. Since no explicit domain information is used, the entire space of state-action pairs must be explored, but this space grows exponentially with the number of sensors. Also, reinforcement learning depends on the ability to detect in which state one is in to map it to the appropriate action. Sensor noise and errors can increase state uncertainty, which slows down the learning process even further.

4.3.2 Other learning methods

Besides reinforcement learning, many other forms of machine learning exist and here we give a short summary of the most common ones.

Case-based learning

With case-based learning, experiences are organized and stored as a case structure, then retrieved and adapted as needed based on the current situation. The basic algorithm is as follows:

1. Classify the current problem.
2. Use the resulting problem description to retrieve similar case(s) from case memory.
3. Adapt the old case's solution to the new situation's specifics.
4. Apply the new solution and evaluate the results.
5. Learn by storing the new case and its results.

Artificial neural networks

An artificial neural network consists of nodes, connected by links that have a certain weight. Learning in artificial neural networks occurs through the adjustment of "synaptic" weights by an error minimization procedure. By increasing the synaptic strength along the neural pathways that are associated with a stimulus and a correct response, frequently used paths are strengthened.

For a good introduction to neural networks see [Aleksander and Morton 1995]. A more in-depth overview can be found in [Hecht-Nielsen 1989].

Genetic algorithms

In genetic algorithms a population of individuals (solutions) is rated using a fitness function and the fittest individuals are allowed to procreate. Since the poorly performing individuals become extinct, over generations the population improves the quality of its set of solutions.
For more on genetic algorithms see [Goldberg 1989]. A good introduction to evolving computer programs (genetic programming) see [Koza 1992].

More on other machine learning techniques can be found in [Mitchell 1997].

4.4 User modeling

User modeling consists of constructing, maintaining and exploiting explicit representations of an individual user or group of users. A **user model**[2] contains information about a user (group) that the system believes to be true. The type of stored information depends on the application. By using knowledge about a user's goals, plans, preferences, beliefs, etc., an IUI can adjust its functionality to better suit the needs. For example, if the user model contains what knowledge a user already possesses, it can improve its information presentation and only provide the user with new and relevant information about the possible actions.
One of the first people who addressed user modeling was Elaine Rich. In one of her papers, she proposed the use of **stereotypes**, which are clusters of related user characteristics [Rich 1979]. Rich built one of the first recommender systems called Grundy using stereotypes. Grundy acts as a librarian and recommends books to its users by asking questions about the user's likes and dislikes.
In the 1980s research in user modeling mainly focused on theoretical issues, most of them from AI. Research topics included methods for making inferences from the user's interactions, user knowledge representation, and reasoning with incomplete information. Several research prototypes were built to demonstrate these theoretical approaches, but the practicality of the application was often forgotten. In the early 1990s, user modeling became more mature and researchers began to look at the possibilities for exploiting user models commercially. Now the empirical aspects of user modeling received more attention, such as user plan recognition, observing the user's interaction with a system, and evaluating the results.

4.4.1 Creating a user model

Before you start designing and implementing a user model into a system, you must ask yourself if it is appropriate for your application. The system should be used by a heterogeneous group of users and have a certain amount of flexibility in its operation. If this is not the case, there is not much point going into all the extra trouble.
The core of any user modeling system is the model of the individual user, also called user synopsis or user profile. A user model is built by combining the information provided by the user, direct inferences from the user's actions, and predictions made by stereotypes or user group models that are believed to be appropriate for this user. Most information in a user model will not be known for sure, since user information often is deduced probabilistically. Therefore, we use ratings for each stored fact that represent how confident the system is that that information is true. We also need to know how we came to this conclusion, so we need some sort of justification why we have attributed this fact to the user. So, a user model will consist of multiple [attribute, value, rating, justification] quadruples, for example [skilled-typist, true, 70%, 321 keystrokes over 1 minute period].

[2] Note that in traditional software-engineering methodologies as well as in much of the HCI literature, a "user model" refers to the a priori constructed designer's model of a user. These models are static and based on the "average user". In IUIs the user model refers to dynamic knowledge about a specific user or group of users which is updated at run-time.

Basically, there are two ways of extracting information to built a user model. Either you automatically try to infer user habits or you simply ask the user for his preferences. In the first case an intelligent system will monitor the user's interaction with the system and try to automatically learn preferences, goals or other interesting user profile data. Although asking the user is much simpler, the advantages of automatically creating user profiles is that the system does not need to bother the user, and changes can be detected automatically.

Asking the user

Asking the user to supply information is the most simple method to gather information for a user model. Unless the user makes a mistake in its answer you do not run the risk of storing false information. The disadvantages are that it takes time for the user to answer the questions and people are not always a good source of information about themselves. For example, people tend to give socially acceptable or desirable answers. Also, it is possible that a user may not be able to answer a question because he does not know the answer. Therefore, a user model should not rely to heavily on the answer of an individual to a specific question.

Learning user characteristics

In order to avoid the disadvantages mentioned above, most user models are constructed by making direct inferences from a user's behavior to a model of the user. User stereotypes can provide a good basis to start from. For example, if we know that someone is a computer scientist, that person will probably type reasonably fast, be higher educated, know how to work with computers and will be able to learn very quickly. Not all of these characteristics may be true, but usually most of them will be. We can assume that the mentioned characteristics are valid until evidence to the contrary presents itself. In order to use stereotypes the system must know what kind of stereotypes there are, what its characteristics are, and how to place a person in one of the stereotypes. The actually used stereotypes and characteristics that make them up will be dependent by the domain and purpose of the system.

A disadvantage of automatically inferring user characteristics is the long period that is needed to gather enough information to make sound deductions. A difficulty that contributes to this problem is that it is very hard to make inferences based on the available input. Usually the only data that is available are mouse clicks and keystrokes. In addition, most operating systems make it difficult to monitor these user interactions. Therefore systems that extract the user model from a user's behavior must take into account that that some of the information is incorrect and can be conflicting with earlier information. Conflicts can also arise when short-term knowledge becomes outdated, so depending on the type of data the system must also make an estimate of the validity period of the stored data.

For user modeling to be most effective, individual user models need to be reused across applications. In the ideal situation the operating system will keep track of the user model and all an IUI needs is to do is to retrieve the information and adapt based on the data in the model. However, a number of problems still need to be solved in order to realize this. First user model applications are far from perfectly due to errors in inferences. Second, the privacy of the user should be guaranteed so other people or programs will not abuse the information in the user model (see also section 2.5.6). Third, we need a standard to store and retrieve the information in a user model. Fourth, it is still unclear what information is most useful to record. Different applications may require different information.

4.4.2 Applications

Microsoft's Lumiere project

The Lumiere project is a project of the Decision Theory and Adaptive Systems Group of Microsoft's research laboratories. One of the main goals of the project was to develop an architecture for reasoning about the goals and needs of software users as they work. The problem addressed by the Lumiere project is how to determine from the available evidence what the user is trying to do at a particular point in time, and then what sort of advice to offer.

The "intelligence" in Lumiere is based on Bayesian user models that capture the uncertain relationships between a user's goals and needs, observations about the current program state, a representation of sequences of actions over time, and the words in a user's query (if a query was made). Events are monitored using an event system that combines atomic actions into higher-level modeled events. The system generates a probability distribution over areas that the user may need assistance with and computes the likelihood that the user would wish to be offered some assistance. The Lumiere project has provided the basis for the Answer Wizard in MS Office 95 and the Office Assistant in MS Office 97.

4.5 Plan recognition

Plan recognition can be seen as a part of the user modeling process. **Plan recognition** is the task of ascribing intentions about plans to an actor (the user), based on observing the actor's actions or utterances [Wærn 1996]. Technically, an operating system processing keystrokes entered by the user is a (very simple) form of plan recognition. The user shows its intentions to the system by pressing keys on a keyboard. However, with plan recognition we normally try to determine more sophisticated and "higher level" information, for example that the user is typing a letter. Plan recognition tries to determine why a user performs a certain series of actions (goal) using the current given input and previous interactions of the user with the system.
Plan recognition draws on several sources of knowledge

- *User input*; the most important information is the input provided by the user or any other information obtained from observing the user by the system.
- *Previous interactions*; the system can make guesses based on previous user input.
- *Common knowledge*; about the user or the world can supplement the other data sources allowing the system to make more accurate presumptions.

4.5.1 Forms of plan recognition

We can distinguish three forms of plan recognition [Wærn 1996]:

1. *Intended plan recognition*; the user is aware and actively co-operating with the recognition system.
2. *Keyhole plan recognition*; the user is unaware or indifferent to plan recognition.
3. *Obstructed plan recognition*; the user deliberately tries to obstruct the plan recognition system (e.g. performing irrelevant actions to confuse the system).

In some cases, intended plan recognition is trivial, for example using a programming language or command-line interpreter to process the user's actions. In other cases such as programming by example (see also section **Error! Reference source not found.**) or meaning extraction in speech processing (section 3.2), things are a bit more complicated. On the average computer system only keystrokes, mouse movements and mouse clicks are available. Sometimes also sound (speech) or images from a camera can be used as input, but these are very difficult to process. In other words, the plan recognition system has only a very small view (keyhole) to look at the user and his intent. In addition, the available information is at a rather low-level compared to the goal we would like to recognize. Besides the limited and low-level information problem, keyhole plan recognition is further complicated because users can try to achieve their goal in different ways or by performing necessary steps in an unexpected order. People are known to use programs in a surprisingly creative manner that the designer did not think of. Since keyhole plan recognition is already very hard to do, obstructed plan recognition is even more difficult, if not impossible.
Plan recognition, and especially keyhole plan recognition, must take into account that user intentions may change over time, or that the user has changed his current goal and has started a completely new task. Therefore any plan recognition system must be careful not to "over-commit" themselves to one possible intention. Especially in a domain where the user can change his goal rapidly, it is often a good idea for the PR system to be a little "forgetful" [Wærn 1996]. Not only

does this reduce the problem of determining when a user changes his intention, but it also makes the system less stubborn. This type of plan recognition that tries to detect a plan based only on a limited set of recently observed actions is also called intention guessing.

4.5.2 Applications and research

In literature, most applications focus either on intelligent tutoring or on intelligent help, two similar applications. With intelligent tutoring, the system tries to help a user to learn some domain knowledge. An intelligent help system usually is less in the foreground then an intelligent tutoring system, but only steps in to help a user if it detects that the user needs help.

Annika Wærn [1996] has designed a reasoning framework for plan recognition in which keyhole and intended plan recognition can be integrated. She states that it is best to combine both keyhole and intended plan recognition. The reason for this is that keyhole plan recognition alone is often too complex and error prone. Since the goal of plan recognition is to help the user, the user can assist the recognition system if it fails and explicitly show his intentions to get help from the system. The drawback however is that the plan recognition system must be designed in such a way that the user can access and inspect it and understand when the system fails. The system must actively provide feedback about its deductions, so in some cases this approach might not be applicable.

Chapter 5: Intelligent interface agents

An intelligent interface agent can be seen as a method of communication with the user as well as a system that learns from the user. Thus it provides a bridge between the previous two chapters dealing with reasoning, learning and different forms of input and output. In the first section we explain what an interface agent is. Next, in section 5.2, we discuss some of the advantages and disadvantages of using anthropomorphic agents. Finally, in section 5.3, we discuss some examples of agents systems.

5.1 What is an intelligent interface agent?

In the last decade the term "agent" has become a buzzword in the computer industry and intelligent interface agents have become a popular research topic in IUIs. However, the overuse of term agent has resulted in many simple non-intelligent agent programs that do not deserve the name. The fact that there is no agreed upon definition of an intelligent agent has very much contributed to this. Despite these difficulties, we will try to explain the general concept.

An **intelligent agent** is a computerized entity that can perform a particular task that has been assigned to it. Usually intelligent agents are autonomous so they can perform this task without any explicit instruction from the user that controls them. This makes them capable of acting on behalf of the user. Another aspect of an intelligent agent is that they are (very often but not always) capable of learning. This learning capability allows them to adapt to new situations they encounter. In addition, they can learn how a user would like them to act. There are many different types of agents ranging from "real-life" robotic agents to virtual interface agents.

An intelligent **interface agent,** also called software agent or softbot, cooperates with the user to accomplish its task and functions as the user's personal assistant. The agent can take the initiative, rather than passively wait for instructions. It can provide the user with information, or detect and correct the user's misunderstandings. Despite the name, an interface agent is usually not the interface between the user and the application. Instead, it observes the interaction between the user and the program from the sideline, learns from it, and interacts with both the user and the program. The interface agent only forms a small part of the whole user interface. Most interface agents create a model of the user and his preferences to provide personalization.

A well-known example of an interface agent is the Office assistant (e.g. Clippy the paperclip) that is found in Microsoft's Office 97 en 2000 programs. The Office assistant can take some initiative and proactively provide information. However, most users who have encountered this agent find it rather annoying. Since a good intelligent user interface needs to perform better than a traditional interface, the Office assistant is not a very good example of an intelligent interface agent.

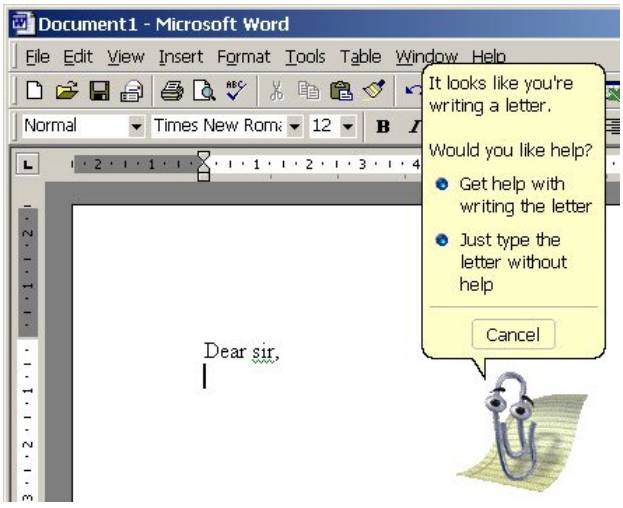

Figure 7: the infamous Clippy the paperclip agent, one of Microsoft's Office assistants

Intelligent interface agents combine many of the earlier discussed topics. Research on interface agents includes dialogue control, user modeling, plan recognition, displaying discourse information via facial displays, multimodal communication, and standards and open architectures for building agent-based interfaces.

5.2 Anthropomorphic agents

Human-human communication is much more effective and comfortable than most forms of human-machine communication. Therefore, many interface agents mimic human behavior, displaying a human-like "personality" and trying to resemble a conscious and intelligent being.
The advantage of using these anthropomorphic interface agents is that they provide a natural and intuitive way of interaction. Agents are very flexible and can help the user learn something by volunteering information. However, the disadvantages are that it is usually not clear to the user which capabilities an agent possesses and what the results will be of the user's input. Since they resemble a human being, for the user it is only logical that it can do the same things humans can do.

A couple of years back there was a discussion about software agents versus direct manipulation [Shneiderman and Maes 1997]. According to Shneiderman, who coined the term "direct manipulation" we should focus on the human capabilities in the visual domain. He proposed to make more use of information visualization. The user is presented with "the big picture" and can zoom in on what they want or filter out what they do not want. By using information visualization in combination with direct manipulation the user feels in control of a predictable and understandable system so that the user can be and feel responsible for his actions. He also argues that the user interface research community should pay more attention to scientific issues such as learning time, performance speed, rate and distribution of errors and subjective user satisfaction.

Table 1: direct manipulation versus software agents

Direct manipulation	**Software agents**
Relies very much on visual representation	Can use different kind of techniques
Direct control and predictability	Indirect control, can have problems with predictability
Users is responsible	Responsibility sometimes less clear
User always takes initiative	Agent is proactive
	Agent can be always running, even when application is closed or stopped.
Unable to learn	Able to learn (adaptive)
Static application or data	Dynamic application or data

Pattie Maes argued in response that software agents allows for programs to be personalized. The agent knows the users habits, preferences and interests. This agent can then be proactive and work for the user even when he or she is not working at the computer anymore. Also agents are adaptive and will adapt to changing user preferences. This is useful because there are an increasingly number of novice people and software systems are becoming more and more complex and dynamic. Also there is an increasingly number of tasks people do on the computer. And when there is a situation of cognitive of information overload one has to delegate.
Recently, there has been more consensus about this topic. Most people now agree that the techniques are complementary and that we need a mixture of the two. For example, using a DM interface for retrieving email and using the agent in that program to delegate some tasks to (e.g. filtering or archiving the email). The initial believe of anthropomorphic agents in the agent community has somewhat been tempered. Agents are not necessarily personified or anthropomorphized.

5.3 Applications

5.3.1 UCEgo; A help system

David Chin has created an Unix consultant called "UCEgo" [Chin 1991]. Using natural language and modeling user goals, and plans, UCEgo can take the initiative in offering the user information about certain Unix concepts or commands, or it can correct the user when it notices misconceptions. UCEgo is even programmed to refuse unethical request such as deleting another user's files.

5.3.2 MOKSAF; Using multiple agents

An example of a multi-agent system for user interfaces is the MOKSAF environment [Payne et al 2000]. The MOKSAF environment is a system for military mission planning and designed to explore teamwork within heterogeneous human/agent teams. A number of interface agents present situation information to the human team members, provide tools and communication methods to create and communicate mission plans.

5.3.3 iBots; Interfacing with existing applications

Traditionally interface agents are designed for a specific application. Either the application is built to work together with the agent, or the agent is programmed to use the programs API. A recent trend is to design agents that can learn to interact with existing application, either through an API or by processing the graphical interface just as a human does. Robert St.Amant even coined a new agent type for the latter; iBot which stands for interface softbot. "*An iBot controls an interactive system through the graphical user interface, as human users do, without relying on an applications programming interface or access to source code*" [St.Amant 2000]. St.Amant's iBots use image processing to retrieve visual interface information and perform actions by adding mouse- and keyboard events to the operating system's event queue. St.Amant proposes to use this type of agent not only as an assistant for users, but also for usability testing to imitate the naïve user.

5.3.4 Open Sesame!

Open Sesame! was one of the first commercially available personal assistant agents created by Charles River Analytics. Its job is to take over repetitive tasks. It observes user activity to look for these repetitive tasks. It can discriminate between events occurring at a particular moment (e.g. you start mail always around 9:00), or event-based (you always open a particular folder when you start your email). If Open Sesame has found a pattern it suggest the user to automate this, the user can then tell the program never to suggest this again, or to accept and possible fine-tune the suggestion.
Michelle Hoyle and Christopher Lueg [1997] report about their experiences working with Open Sesame! for about nine months. During this period, Open Sesame! made 129 different suggestions, but only 2 were immediately accepted. Hoyle and Lueg claim that the main reason that Open Sesame! does not live up to its expectations is because it does not make use of situatedness. With **situatedness** (also referred to as context awareness) they mean that the user can be in a number of different situations when performing an action. The exact action that should be taken by an agent should be dependent on the current situation (context).

Chapter 6: Future visions

In this last chapter, we will provide the reader with some ideas about the current interface developments and future applications. In section 6.1, we discuss the current direction of IUI research. Then, we will look at the future plans and ideas for multimodal interfaces in section 6.2. Finally, we will provide some personal remark about the ideal interface.

6.1 Where do we go from here?

Although undoubtedly, many improvements will be made to the current object-oriented direct-manipulation interfaces that we use now, it is very likely that interface of the future will be very different. As we mentioned earlier in this report, more and more devices are being computerized and each computerized device is becoming more intelligent, hiding as much of the functionality from the user as possible.
It is doubtful that all these devices will use a common and generally accepted interface and interaction style. Instead, we already see that for each device the interface is specifically designed to fit the application, the environment, and the individual users' needs. This way, next-generation interfaces allow users to focus on their task rather than operating a device. The user provides a high level description of the task he wants to delegate, and the machine figures out the details. For example, recently smart VCR's have been introduced that only require you to enter the names of your favorite television shows. Once entered, the VCR checks when each show is on the air and records it for you, automatically skipping the commercial breaks. It is no longer is necessary to find out and enter the start time and end time of television shows you want to tape.
However, the debate about how much automation is desirable is still going. The question still is what responsibility do we give to the machine and what would we like to do ourselves.

In the coming years, several problems need to be solved and tools have to be create before we can create commercially acceptable and useful IUIs [Höök 2000]:

- A better understanding of how and when *useful (general) adaptation techniques* can improve the interaction (i.e. design practice);
- Reliable and cost-efficient *IUI development methods*;
- *Authoring tools* that enable easy development as well as maintenance of the intelligent parts of the system.
- *Usability principles* that are specifically designed for intelligent interfaces (rather than DM systems);

These problems will not be solved overnight. Several disciplines need to contribute, human-computer interaction, artificial intelligence, psychology etc.

6.2 Multimodal interfaces

The field of multimodal interfaces is a very promising one. With sensor technology becoming more reliable and cheaper to use, multimodal interfaces will be introduced as a new way of interfacing not only at work or at home, but also on the road via cell phones or PDA's. Whereas early multimodal systems focused on active communication through natural language processing and gestures, recently more attention has been given to passive communication through vision-based systems (gaze tracking and facial expression recognition). It is expected that blended interface styles, a combination of active and passive interfaces, will get more attention. As computing power increases and additional input channels become available, intelligent interfaces will be more context-sensitive, making complex inferences about the user's goals using several information sources. The future interface will assess the emotional state of the user with image processing and voice recognition technology. Biometric measures, such as fingerprint or eye scanning, will be used to identify people.

At the moment, a large drawback of most multimodal interfaces is that it requires rather intrusive hardware such as gaze-trackers, gloves and/or helmets with positioning sensors that get the necessary data.. Recently advanced multimodal architectures have been devised that are capable of incorporating several modalities. However, more knowledge is needed about the natural interaction methods of humans, how to effectively combine modalities, and how the use of one modality can influence others. [Oviatt et al 2000] has identified a number of future research directions for multimodal interfaces:

1. *Cognitive theory and empirical science underpinnings*. A better understanding is required of the unique linguistic and performance characteristics of natural communication modalities (speech, gesture, gaze and facial expressions) and of how these modalities can be combined best.

2. *New multimodal interface concepts*. Current research on multimodal interfaces has focused mainly on natural language processing, pen-based gestures. Additional input such as body motion and facial expressions should be studied.

3. *Multimodal language and dialogue processing*. A general theory of conversational interaction is needed that deals with intent representation for non-speech modalities.

4. *Error handling techniques*. Graceful error handling is still a problem in multimodal interfaces and mutual disambiguation between signals can be improved. Also the impact of a third or more modalities on the error rate should be studied, as well as performance under difficult (mobile) environments.

5. *Adaptive multimodal architectures*. Multimodal architectures are hardly adaptive and adapting to a specific user or environment can increase the recognition of input processing, as well as provide more flexibility and ease of use for the user.

6. *Multi-device multi-user ubiquitous computing.* In the future, mobile computing will become more important, so we need to look at the role of multimodal interfaces in that area. In addition, when multiple users are interacting together, for example via the Internet, interfaces need to take multiple input from remote devices into account.

7. *Multimodal research infrastructure.* Supporting tools for multimodal interface research should be developed. This includes; semi-automatic simulation methods for empirical data collection and prototyping of new systems, automated tools for collecting and analyzing multimodal corpora, novel metrics for evaluating multimodal systems, and software tools that support the rapid creation of next-generation multimodal applications.

6.3 Final remarks

Currently, we can still see a distinction between multimodal systems, adaptive systems and wireless systems, but the differences are already becoming smaller and will soon fade. Multimodal systems are starting to become adaptive and more technology is becoming wireless.
Personally, I think that there will be a limit how far you can go with automation things. In the end, we must be in control and the computer will remain a tool or assistant designed to help us. Also, there is still a need for reflection, thinking and planning, so interfaces do not always need to act. The ultimate computer interface is something like in Star Trek; engaging in an dialog with the computer, or even the holodeck, an advanced virtual reality environment you can interact with. But just like in Star Trek, it's not always ideal for all purposes. However, we are still a long way off from the ideal interface, if any such thing exists.

Bibliography

Aleksander, I. and Morton, H. (1995) "*An introduction to neural computing",* (2nd edition), International Thomson Computer Press.

Ball, G., Ling, D., Kurlander, D., Miller, J., Pugh, D., Skelly, T., Stankosky, A., Thiel, D., Van Dantzich, M. and Wax T. (1996) "Lifelike computer characters: the Persona project at Microsoft research" in *Software Agents*, Bradshaw, J.M. (editor), AAAI/MIT Press, Cambridge, MA, 1996, pp. 191–222. Also available as http://research.microsoft.com/research/ui/persona/chapter/persona.htm (February 26, 2003).

Benyon, D. (1993) "Adaptive systems: a solution to usability problems" in *User Modelling and User-Adapted Interaction, Vol. 3*, No. 1, pp. 65-87.

Birnbaum, L., Horvitz, E., Kurlander, D., Lieberman, H., Marks, J. and Roth, S. (1997) "Compelling intelligent user interfaces; how much AI?", in *Proceedings of the 1997 international conference on Intelligent User Interfaces* (IUI'97), panel discussion, pp. 173-175, Orlanda, FL, USA. Also http://www.merl.com/reports/docs/TR96-28.pdf (February 26, 2003).

Bolt, R. (1980) "Put-that-there: voice and gesture at the graphics interface.", in *Computer Graphics Vol. 14*, Issue 3, pp. 262-270, July 1980, (ACM SIGGRAPH Proceedings). Also in *Reading in intelligent user interfaces*, Maybury, M.T. and Wahlster, W. (editors). 1998, Morgan Kaufmann Publishers, San Francisco, CA, USA.

Brookings, J.B., Wilson, G.F. and Swain, C.R. (1996) "Psychophysiological responses to changes in workload during simulated air traffic control", in *Biological Psychology, Vol.42*, No.3, pp. 361-377, Elsevier Science.

Chin, D. (1991) "Intelligent interfaces as agents.", in *Intelligent user interfaces,*.Sullivan, J. and Tyler S. (editors), ACM Press, New York, USA.

CLIPS (2003) The Official Home Site of CLIPS http://www.ghg.net/clips/CLIPS.html (February 26, 2003)

Cook, R. and Kay, J. (1994) "The justified user model: a viewable, explained user model", *in Proceedings of the 4th International Conference on User Modeling*, pp. 145-150, Hyannis, MA, USA. http://citeseer.nj.nec.com/cook94justified.html (February 26, 2003).

Ekman, P. and Friesen, W.V. (1978) "*Facial Action Coding System (FACS): manual*", Consulting Psychologists Press Press, Palo Alto, USA.

Engelbart, D.C. and English, W.K. (1968) "A research center for augmenting human intellect", *AFIPS Conference Proceedings of the 1968 Fall Joint Computer Conference*, San Francisco, CA, USA, December 1968, Vol. 33, pp. 395-410.

Extempo (2002) http://www.extempo.com (February 26, 2003).

Jacob, R.J.K. (1991) "The use of eye movements in human-computer interaction techniques: what you look at is what you get.", *in Transactions on Information Systems Vol. 9*, No. 3, pp. 152-169, April 1991, ACM Press. Also in *Readings in Intelligent User Interfaces,* Maybury, M.T. and Wahlster, W (editors), 1998, Morgan Kaufmann Publishers, San Francisco, CA, USA.

Goldberg, D.E. (1989) "*Genetic algorithms in search, optimization and machine learning*". Addison-Wesley, Reading, Massachusetts.

Hecht-Nielsen, R. (1989) "*Neurocomputing*", Addison-Wesley Publishing Company, Inc.

Höök, K. (2000) "Steps to take before intelligent user interfaces become real.", in *Interacting with Computers; the interdisciplinary journal of Human-Computer Interaction Vol. 12*, Issue 5, pp. 409-426. Also http://www.sics.se/~kia/papers/Steps_Hook_final.pdf (February 26, 2001).

Horvitz, E (1999) "Principles of mixed-initative user interfaces", in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (CHI'99), pp. 159-166, Pittsburgh, PA, USA, May 1999. Also http://research.microsoft.com/~horvitz/UIACT.HTM (February 26, 2002).

Jelinek, F. (1999) "*Statistical methods in speech recognition*", MIT Press.

Koons, D.B., Sparrell, C.J., and Thórisson, K.R. (1997) "Integrating simultaneous input from speech, gaze, and hand gestures." in *Readings in Intelligent User Interfaces,* Maybury, M.T. and Wahlster, W (editors), 1998, Morgan Kaufmann Publishers, San Francisco, CA, USA.

Koza, J.R. (1992) "*Genetic programming: on the programming of computers by means of natural selection*", MIT Press/Bradford Books Edition, Cambridge, MA, USA.

Lanier, J. (1995) "Agents of Alienation", in *Journal of Consciousness Studies, Vol. 2*, pp 76-81. Also http://www-cse.ucsd.edu/users/goguen/courses/271/lanier.agents.html (February 26, 2002)

Lieberman, H. (1997) "Autonomous interface agents", in *Proceedings of the ACM conference on Computers and Human Interface* (CHI-97), Atlanta, GA, USA, March 1997.

Maes, P. (1994) "Agents that reduce work and information overload." In *Communications of the ACM, Vol. 37*, No. 7, pp. 31-40, ACM Press, July 1994. Also http://pattie.www.media.mit.edu/people/pattie/CACM-94/CACM-94.p1.html (February 26, 2002).

Maybury, M.T. (2001) "Intelligent user interfaces for all.", in *User interfaces for all: concepts, methods, and tools.*, Stephanidis, C. (editor), Lawrence Erlbaum Associates, Inc., Publishers, Maway, NJ, USA.

Maybury, M.T. and Wahlster, W. (1998) "*Readings in intelligent user interfaces*", Morgan Kaufmann Publishers, Inc., San Fransico, CA, USA.

Mitchell, T.M. (1997) "*Machine learning*", McGraw-Hill Companies, Inc.

Merriam-Webster (2003) "*Online collegiate dictionary*", http://www.m-w.com (February 17, 2003).

Meystel, A.M. and Albus, J.S. (2002) "*Intelligent systems: architecture, design and control.*", John Wiley & Sons, Inc., New York, USA.

Myers, B., McDaniel, R. and Wolber, D. (2000) "Intelligence in demonstrational interfaces." in *Communications of the ACM, Vol. 43*, No 3, pp. 82-89, March 2000.

Nagel, K., Kidd, C.D., O'Connel, T., Dey, A., Abowd, G.D. (2001) "The family intercom: developing a context-aware audio communication system.", in *Ubicomp 2001, LNCS 2201*, pp. 176-183, Abowd, G.D., Brumitt, B. and Shafer, S.A.N. (editors), Springer- Verlag, Berlin Heidelberg.

Neal, J.G., Thielman, C.Y., Dobes, Z., Haller, S.M., Shapiro, S.C. (1997) "Natural language with integrated deitic and graphic gestures.", in *Readings in Intelligent User Interfaces,* Maybury, M.T. and Wahlster, W (editors), 1998, Morgan Kaufmann Publishers, San Francisco, CA, USA.

Oviatt, S., Cohen, P., Wu, L., Vergo, J., Duncan, L., Suhm, B., Bers, J., Holzman, T., Winograd, T., Landay, J., Larson, J. and Ferro, D. (2000) "Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions" in *Human-Computer Interaction, Vol.15*, No 4, pp. 263-322, December 2000.

Pantic, M. (2001) "*Facial expression analysis by computational intelligence techniques*", PhD. thesis, Knowledge-based systems group, Delft University of Technology, The Netherlands.

Preece and Shneiderman (1995) "Survival of the fittest: the evolution of multimedia user interfaces", in *ACM Computing Surveys, Vol. 27*, No.4, December 1995.

Rabiner, L.R., Juang, B.H., (1993) "*Fundamentals of speech recognitio*n", Prentice Hall, NJ, USA.

Rich, E. (1979) "User modeling via stereotypes." in *Cognitive science, Vol. 3*, pp. 329-354.

Rime, B. and Schiaratura, L. (1991) "Gesture and speech", in *Fundamentals of Nonverbal Behaviour,* Feldman, R.S. and Rime, B. (editors), pp. 239-281, Cambridge University Press.

Russell, S. and Norvig, P. (1995) "*Artificial Intelligence: a modern approach"*. Prenctice-Hall International, Inc.

St.Amant, R. (2000) "Interface agents as surrogate users." in *Intelligence; new visions of AI in practice, Vol. 11*, No. 2, pp. 28-38, summer 2000.

Shneiderman, B. (1997) "Direct manipulation for comprehensible, predictable and controllable user interfaces", in *Proceedings of 1997 International Conference on Intelligent User Interfaces*, ACM, Orlando, FL, USA, January 1997.

Shneiderman, B. (1997) "*Designing the user interface: strategies for effective human-computer interaction*.", Addison-Wesley Publishing Co., Menlo Park, CA, USA.

Shneiderman, B. and Maes, P. (1997) "Direct manipulation vs. interface agents.", in *Interactions, Vol. 4*, Issue 6, pp. 42-61, ACM Press.

Suchman, L.A. (1997) "From interactions to integrations: a reflection on the future of HCI", *Keynote address at the 6th IFIP International Conference on Human-Computer Interaction* (INTERACT 97), Sydney, Australia, July 1997. Available as http://www.acs.org.au/president/1997/intrct97/suchman.htm (March 7, 2002).

Suchman, L.A. (1987) "*Plans and situated actions; the problem of human-machine communication*", Cambridge University Press.

Sutherland, I. (1963) "*Sketchpad: a man-machine graphical communication system*", PhD. thesis, Department of Electrical Engineering, Massachusetts Institute of Technology. Available as http://theses.mit.edu:80/Dienst/UI/2.0/Describe/0018.mit.theses/1963-10 (February 26, 2003).

Svensson, E., Angelborg-Thanderz, M., Sjoberg, L. and Olsson, S. (1997) "Information complexity-mental workload and performance in combat aircraft" in *Ergonomics, Vol. 40*, No. 3, pages 362-380, March 1997, Taylor and Francis Ltd.

Tecuci, G. (1998) *"Building intelligent agents: an apprenticeship multistrategy learning theory, methodology, tool and case studies*". Academic Press, San Diego, CA, USA.

Turing, A.M. (1950) "Computing machinery and intelligence", in *Mind: a quarterly review of psychology and philisophy, Vol. LIX*, No.236, October, 1950. Also available as http://www.abelard.org/turpap/turpap.htm (February 26, 2003).

Vark, R.J. van (1997) "*Designing dialogue management in the Alparon project*", Alparon report 97-01, Knowledge Based Systems group, Delft University of Technology, The Netherlands. Available as http://www.kbs.twi.tudelft.nl/Publications/Report/1999_and_earlier/1997-VanVark-ARS-01.html (February 26, 2003).

Venkatesh, A., Shih, E.C., Stolzoff, N.C. (2000) "A longitudinal analysis of computing in the home census data 1984-1997", in *Proceedings of the International Conference on Home Oriented Informatics and Telematics* (HOIT 2000), June 2000, Wolverhamption, UK.

Wærn, A. (1996) "*Recognising human plans: issues for plan recognition in human-computer interaction*", PhD thesis, Department of Computer and Systems Sciences, The royal institute of Technology and Stockholm University, Sweden, May 1996.

Wojdel, J. C. and Rothkrantz, L.J.M. (2001a) "Robust video processing for lipreading applications", in *Proceedings of 6th annual scientific conference on web technology, new media, communications and telematics theory, methods, tools and applications* (EUROMEDIA 2001), Valencia, Spain, April 2001. Also http://www.kbs.twi.tudelft.nl/Publications/Conference/2001/ (February 26, 2003).

Wojdel, J. C. and Rothkrantz, L.J.M. (2001b) "Using aerial and geometric features in automatic lip-reading", in *Proceedings the 7th European Conference on Speech Communication and Technology* (EuroSpeech 2001), Aalborg, Denmark, September 2001. Also http://www.kbs.twi.tudelft.nl/Publications/Conference/2001/ (February 26, 2003).

Wyard, P.J., Simons, A.D., Appelby, S., Kaneen, E., Williams, S.H., Preston, K.R. (1996) "Spoken language systems-beyond prompt and response", in *BT Technology Journal, Vol. 14*, No. 1, pp. 187–205. Available as http://www.csd.abdn.ac.uk/~swilliam/Publications/BTTJ1996.pdf (February 26, 2003).