1

Introduction

1.1 Two Metaphors for Interaction Design

1.1.1 The Computer as a Tool

The world around us is changing rapidly and we are increasingly surrounded by electronic devices which provide us with various types of information. Internet and mobile applications are commonplace and future predictions describe a world where computers are embedded increasingly in the environment in which we live and in the ways in which we work: in a few years' time, for instance intelligent home robotics will cater for our many everyday needs, and we will be required to interact with a complex environment which will consist not only of people, but also of computers embedded in our daily surroundings. Technological roadmaps have envisaged how smart environments will be populated by several context-aware devices which communicate with each other and with the users, and how the future information and communication systems will contain computers built into products such as clothes, books, beds and sporting gear (Sjöberg & Backlund, 2000; Plomp et al., 2002). The systems will identify their current context of use, adapt their behaviour and allow for natural interaction. Computers will also have senses and be able to interpret human expressions, will be able to smell, feel, hear, see and taste, and there will be intuitive human-computer interfaces that mimic human communication.

In such environments, our interactions with the computer will also become more complex. The internet provides an information source as well as an infrastructure for embedded computing and virtual interaction, while mobile communication and wireless appliances can be deployed for building various digital services. Many of our everyday tasks already require the use of a computer as a tool: text editing, image processing, communication with friends and colleagues by emails and web-phones, information seeking in the internet and in various digital databases. Computers also enter into human activities where they replace the other participant: interactive situations with automatic services are commonplace, and different types of online services (news, banking, shopping, hotel and flight

reservations, navigation, e-government and e-administration) are in extensive use. Moreover, computer agents can also act on behalf of the user when given the user's identity in simulated worlds, but also in the real world as e.g. taking care of automatic payments or news watching when assigned the user's requirements and preferences.

Often these interactions presuppose communication in some form of natural language: information seeking and question answering, route navigation and way finding, tutoring, entertainment and games, dialogues with robots and talking heads, not to mention human-like conversational settings such as negotiations, monitoring of meetings and social chatting all involve communication using natural language words. Moreover, interactions with computers are not necessarily conducted through graphical interfaces with the keyboard and mouse, but can also include multimodal aspects such as speech, maps, figures, videos, and gesturing, and the screen need not only include text boxes and windows but also animated agents and talking heads which create the illusion of a fellow agent.

So far the general view of human computer interaction has regarded the computer as a tool: it supports human goals for clearly defined tasks (see e.g. Shneiderman, 1998). Interface design has focused on designing easy-to-use artefacts, and the main research activities are concerned with usability issues, i.e. how to balance task requirements and human factors so as to assist the user to perform the intended task as efficiently as possible. Main applications have dealt with graphical user interfaces (GUI) and web portals, as well as assistive tools for such complex tasks as computer-aided design and computer supported collaborative work where the interface is intended to support the users' work and thinking. The emphasis on the interface design has been on clarity, simplicity, predictability and consistency, which make the interface easy and transparent to use. Moreover, the user should have control of the properties of the interaction, i.e. of the system's adaptive capabilities.

The same principles have also been applied to the design of natural language interfaces. Speech is often regarded as an alternative mode that can be fixed on top of GUI-type interfaces in order to read out menu commands or to give a more natural sound to dialogue type interfaces. The main emphasis is thus on system responses which should be geared towards clear, unambiguous prompts. The prompts should take the user's cognitive limitations into account and provide the user with explicit information about the task and the system's capabilities (see e.g. Dix et al., 1998; Weinschenk & Barker, 2000). In the development of speech-based applications, the emphasis has also been on a high degree of accuracy of speech recognition and word-spotting techniques so as to pick the important concepts from the user input. The so-called How May I Help You (HMIHY) technology (Gorin, Riccardi & Wright, 1997) has been influential in this respect, and helped to build applications where, typically, the user can reach her goal after a few spoken exchanges. To minimize the speech recognition errors, the user is given only a couple of choices to choose the response from, and the interaction is strictly system-directed, i.e. the dialogues are driven by the system questions.

Speech has often been regarded as an alternative mode that can be fixed on top of GUI-type interfaces in order to read out menu commands or to give a more natural sound to dialogue interfaces. However, as pointed out by Yankelovich (1996), speech applications are like command line interfaces: the available commands and the limitations of the system are not readily visible, and this presents an additional burden to the user trying to familiarise herself with the system. The heart of effective interface design thus lies at the system prompt design as it helps users to produce well-formed spoken input and simultaneously to become familiar with the functionality that is available. Yankelovich herself gives best-practise guidelines in her SpeechAct system, and introduces various prompt design techniques to furnish the system with natural and helpful interface. Some such techniques are tapering, i.e. shortening the prompts as the users gain experience with the system, and incremental prompts, i.e. incorporating helpful hints or instructions in a repeated prompt, if the previous prompt was met with silence (or a timeout occurs in a graphical interface). The system utterances are thus adapted online to mirror the perceived user expertise.

Spoken language interfaces also set out expectations concerning the system's capability to understand and provide natural language expressions. For instance, Jokinen and Hurtig (2006) noticed that the same system is evaluated differently depending on the users' predisposition and prior knowledge of the system. If the users were told that the system had a graphical-tactile interface with the ability to input spoken language commands, they evaluated the spoken language communication as being significantly better than those users who thought they were interacting with a speech-based system which also had a multimodal facility. A spoken language interface appeared to bring in tacit assumptions of fluent human communication and consequently, expectations of the system's similar communication capabilities were high, making the actual experience disappointing as the expectations were not met. With the graphical-tactile interface, speech was not regarded as the primary means of communication; instead speech provided an interesting extra value to the interaction, and thus was not judged as harshly.

When dealing with the present-day interactive services, however, users need to adapt their spoken communication methods to the those available in the technology. Interactions with natural language dialogue systems are usually characterised by repetitive and unhelpful patterns which are designed to operate in a limited context on a particular task, and to provide easy-to-process input for the system. They leave the users unsatisfied if the usage context is different from that designed in the system. Moreover, the problem is not only that the users may not complete the task that the system is meant to be used for, but that the interaction itself does not show the sensitivity to the users' needs that characterise human communication. In other words, spoken dialogue systems are not only required to meet their task specification, but they should also cater for the user's needs concerning intelligent knowledge management and adaptation to the situation.

Human-human communication involves the smooth coordination of a number of knowledge sources: the characteristics and intentions of the speakers, the topic and focus of the conversation, the meaning and frequency of lexical items, multimodal aspects such as gesturing, facial expressions, and prosody, as well as communicative context, physical environment, world knowledge, etc. The challenges associated with spoken language interface design can be identified as three complex areas: natural language meaning, dialogue strategy, and user adaptation. All of them are related to knowledge management and construction of the shared context. For instance, natural language meaning does not simply function in a compositional manner whereby the semantics of a construction would be the sum of the semantics of its components. Rather, the meaning is constructed via interaction between the item and its context. It can be said that the meaning of an item is the sum of all its contexts (see the computational techniques in Lund and Burgess, 1996, and in Sahlgren, 2005), and thus the meaning of natural language is affected by the contexts in which it is used and shared. Various dialogue strategies are also important when building and managing the shared context. The speakers use natural language to clarify, explain, and describe complicated issues, and they apply their intuitive behaviour patterns, politeness codes, and communication styles to interpret and react to the partner's utterances. Finally, the speakers also adapt to each other and the shared context, and tailor the interaction according to their particular needs. Gradual alignment with the partner takes place on all levels of communication: lexical items, sentence structure, pitch level, intonation, body posture, etc. (see, e.g. the experiments on the speaker alignment in Pickering & Garrod, 2004). The three interface challenges will be discussed more in the later chapters of the book.

We must note that if the features of human-human communication are transferred to spoken interactive systems, the users are involved in an activity that differs from and even contradicts what they are used to when usually communicating with natural language: although the interaction mode is natural language, the interacting partner is a tool and not a fellow human. Since humans are extremely adaptable users, they can learn interaction patterns that are unintuitive in human communication (e.g. wait for a signal to speak, or mark with carriage return or push button to end their turn), and also learn to speak in a clear and articulated manner. However, the very use of natural language as an interface mode also directs the user's actions towards communication rather than simple manipulation of objects and task completion. The computer is treated like a fellow partner in the communicative situation, and natural language interaction seems to inherit the same kind of intuitive interaction patterns that prevail in human-human communication. For instance, the studies by Reeves and Nass (1996) show that the users anthropomorphise, i.e. assign human characteristics to the computer. The users see personality in computers, and assess the personality very quickly with minimal clues. The perceived personality further affects how the users evaluate

the computer and the information it gives to the user. For instance, humans think highly of computers that praise them, even though praise is expressed as obvious flattery and may not be deserved at all. The users also rated the quality of a computer more frankly if the evaluation questionnaire was presented to them on another machine than the one they were meant to evaluate; Reeves and Nass concluded that the users unintentionally avoided hurting the feelings of a computer which they had just interacted with.

People thus seem to equate media with real life, and the computer with a human-like agent. Even though they may not realise that this is what they are doing, the complexity and unfamiliarity of interactive applications seem to give reasons to consider the computer as a communicating agent rather than a controllable tool. Indeed, the future digital environment that is envisaged in the ubiquitous computing paradigm (Weiser, 1991: http://sandbox.xerox.xom/ubicomp/) encourages such interaction: digital services, sensory devices, etc. populate our environment and penetrate into all aspects of life, ready to change our working habits and interaction patterns in a fundamental way. Digital participants are included as other communicating partners, and the appliances are aware of the other applicances as well as the users, they adapt and respond to the users' needs and habits, and are accessible ubiquitously via natural interaction.

New interaction techniques are also being developed to complement and compensate conventional graphical, mouse and keyboard interfaces. Besides speech, technologies for tactile, gesture and eye-gaze interaction have matured in the past ten years, and enable the user's multimodal communication with applications. Taking advantage of the entire repertoire of different media and communication channels, and by mimicking the possibilities available in human communication, these novel techniques aim at supporting richer and more natural interactions with applications. The term *Perceptual User Interfaces (PUI)* has been introduced (Turk & Robertson, 2000) to cover interfaces that combine natural human capabilities (communication, motor, cognitive, perceptual skills) with computer I/O devices, and thus include multimodal communication in the wide sense. Although perception and multimodal communication is natural and intuitive in human communication, many open issues still deal with the design and use of such systems. Especially, a deeper understanding is needed of the manifestations and functionality of the information received via different input channels, and of the relation between verbal and non-verbal communication for the exchange of messages: how do human speech and gestures, gaze and facial expressions contribute to communication, how are the presentation modes and the symbolic content coordinated, interpreted and integrated into meaningful actions? However, within the wide variety of novel interface technology, it is easier to see the system realised as a conversational interface agent that mediates interaction between the user and the application, than as a tool which is used to control and perform certain aspects of a task.

It is also important to survey tasks and applications where communicative systems would be superior to directly controllable systems. For instance, conversing toasters may not be as impressive showcases for spoken dialogue systems as hand-held conversational devices that inform the user about appointments or tourist attractions. Tasks with a straightforward structure may not allow the benefits of flexible natural language conversation to come through: although call routing and voice controlled applications are useful speech-based services, their interaction capabilities need not go past question answering dialogues for which present-day interaction technology already provides management techniques. Instead, tasks that require complex reasoning, evaluation, and integration of pieces of information seem more appropriate for rich dialogue interactions. Indeed, viable spoken dialogue application domains in this respect have also appeared: guiding and training novice users for skill acquisition in tutoring systems, navigation and way finding for location-based services, speaking in games and entertainment, social chatting in human-robot interactions. Besides information search and presentation, these tasks presuppose the participants' ranking of their immediate goals, linking of focussed information to their previous knowledge, and balancing of long-term intentions with the partner's requests. Interactions are not necessarily predictable in advance but depend on the context and actions of the individual participants: communicative situations are constructed dynamically through the participants' communicative (and physical) actions. The driving force for interactions thus appears to reside in the participants' intention to exchange information, while the flow of information is controlled by the participants' awareness of the situation and by their reactions to the contextual changes. Consequently, the tasks require communication of certain intentions and knowledge, negotiation skills, and awaress of the partner's needs, and there appears to be a need for a user-friendly flexible interface that would understand the logic behind natural language utterances as well as that of communicative principles.

1.1.2 The Computer as an Agent

In the past years, a new metaphor for human-computer interactions has thus emerged: the computer as an agent. The computer is seen as an assistant which provides services to the user, such as helping the user to complete a task by providing necessary information of the different steps of the task (e.g. tutoring, explaining, medical assistance), or helping the user to organize information (e.g. electronic personal assistants).

Future interactions with the computer are envisaged to use possibilities offered by multimodal interaction technology and to resemble human communication in that they are conducted in natural language. The new types of interfaces are expected to possess near human-like conversation capabilities so as to listen, speak and gesture, as well as to address the user's emotional state in an intelligent manner.

On the other hand, human control over the system's behaviour is not so straightforward and as easily managed as before. Computers are connected with each other into networks of interacting processors, and their internal structure and operations are less transparent due to distributed processing. Object oriented programming and various machine-learning techniques make the software design less deterministic and the input-output relation less controllable. Context-aware computing strives to build systems that learn the preferences of individual users while also understanding the social context of user groups, adapting decisions accordingly. Simply, the view of the computer as a tool which is a passive and transparent "slave" under human control is disappearing.

It must be noticed, however, that the agent metaphor has been used in three different research contexts which are quite separate from each other. Consequently, the metaphor bears different connotations in these contexts. Below we will discuss the notion of "agent" in interface design, robotics, and software design. One of the obvious uses of the agent metaphor is connected to interface agents, i.e. to various animated interface characters which aim at making the interaction more natural. It is argued that personification of the interface makes interaction less anonymous, and thus contributes to the user's feeling of trust in the system. For instance, in the SmartKom project (Wahlster, 2004), the metaphor was realised in the Smartakus – an interface agent capable of mediating interaction between human users and the application: it took care of the user's requests with respect to different applications and application scenarios. Other examples, and of more human-like animated agents, include the museum guide August (Gustafson et al., 1999), real estate agent REA (Cassel et al., 2001a), Medical Advisor GRETA (Pelachaud et al., 2002a), and Ruth (DeCarlo et al., 2002).

Interface agents equipped with human-like communicative, emotional and social capabilities are usually called Embodied Conversational Agents (ECAs) (Cassell et al., 2003). ECAs have their own personality, different abilities, and they employ gestures, mimics and speech to communicate with the human user, see e.g. Pelachaud and Poggi (2002) for a discussion on multimodal embodied agents. André and Pelachaud (forthcoming) provide an overview of the development of ECAs, starting from TV-style presenters (e.g. ANANOVA, http://www.ananova.com) through virtual dialogue partners such as Smartakus, to role playing agents and multiparty conversational partners. Scenarios where human users truly interact with synthetic agents are still experimental, however. The earlier ones concern immersive virtual environments where the users can interact with the agents, and include such systems as CrossTalk (Gebhard et al., 2003) and VicTec (Paiva et al., 2004).

André and Pelachaud (forthcoming) also point out that the interaction with interface agents creates a social environment where norms and cultural rules are expected to be followed. Most ECA studies explore various social aspects of interaction and experiments are designed to bring in novel insights concerning interaction with synthetic agents. In many cases, the studies concern game

and entertainment environments: e.g. Isbister et al. (2000) studied social interaction between several humans in a video chat environment, while Becker and Wachsmuth (2006) focus on a game scenario in which the agent takes on the role of a co-player.

Interaction management itself is usually based on schemas. However, in the system developed by Traum and Rickel (2002), plan-based dialogue management techniques with dialogue acts turn-taking and grounding are used, and extended in multiparty dialogue contexts. The task concerns rehearsal of military missions, and the dialogues take place between the human user and virtual agent or between virtual agents. The interaction model allows reasoning about the other agents and their intentions, emotions, and motivations, and later developments also include teamwork and negotiation (see e.g. Traum et al. 2003). The agents are thus capable of simulating social interaction, and interesting research topics in this respect concern e.g. cultural and personality differences. Also in the Collagen system (Sidner, Boettner & Rich, 2000), the interface agent is a collaborative agent which implements the SharedPlan discourse theory (Grosz & Sidner, 1990, Grosz & Kraus 1995) and plan recognition using hierarchical task network (Lesh, Rich & Sidner 1998). The framework has also been integrated in animated agents (Cassell et al. 2001).

Another use of the agent metaphor is found in AI-based interaction management. The BDI (Belief-Desire-Intention) agents refer to autonomous intelligent systems that model such cognitive skills as planning and reasoning so as to be able to deliberate on the appropriate action in a given situation. For instance, the last two examples above can also be said to belong to this tradition as they use the classic techniques developed in the context of communicating BDI-agents. A continuation of this research can be found in robotics, although probabilistic and machine-learning techniques have mostly replaced the traditional handcrafted reasoning rules, and novel applications range from intelligent cars to soccer playing robots. Autonomous robots are also called Situated Embodied Agents (SEAs) as they are directly interacting with their environment: the robots are equipped with various sensors and vision components through which they observe their environment and with motor components through which they produce suitable actions. Interaction with the environment is based on rapid reasoning about the appropriate action in a given situation and, if the robots work as a team, on communication with one another, so as to coordinate their collaboration.

If the robots are placed in situations where the task requires human intervention, communication is most naturally conducted in natural language. One of the first examples of such situated dialogues is the robot Godot (Bos et al., 2003), which interacts with the user in a museum environment. Dialogues with mobile and sensing robots also offer a novel research topic for interaction management: language grounding. Although the term grounding is well known in dialogue research, dealing with feedback and construction of a shared understanding as part of successful communication, the challenging aspect with robots is the

opportunity to talk about the physical environment which is shared between the user and the robot. Reference to elements in the environment must be grounded taking into account the participants' vision field and their focus of attention at a given moment, i.e. a relation between the element and its linguistic expression must be established. Kruijff et al. (2007) present an ontology-based approach to the problem in the context of human-robot interaction where a human user guides the robot around a new environment, and the robot incrementally builds up a geometric map of the environment through its interaction with the user. The situated dialogue understanding is established by linking semantic representations of the utterance meaning with the robot's internal world representation. The three-layer world model connects the spatial organisation of an environment with a commonsense conceptual description of the environment, and thus provides the basis for grounding linguistic meanings into the innate conceptual knowledge of the objects encountered by the robot in its environment.

Like ECAs, robots also provide a testing ground for social interaction and expressive face-to-face communication. One of the first to engage people in natural and affective, social interaction was Kismet (Breazeal, 2002), a robotic head which was inspired by a caretaker-infant interaction to explore various social cues and skills that are important in socially situated learning. The communication robot Fritz (Bennewitz et al., 2007) is a humanoid robot, which mimics the way humans interact with each other, and thus aims to give insights of efficient and robust multimodal human communication strategies and their transference to human-machine interface. Fritz uses speech, facial expressions, eye-gaze, and gestures, and it can generate synchronised gestures and speech as well as change its attention between the communicating partners in the vicinity. The results of its public demonstrations showed that people found the robot human-like and an enjoyable communication partner. A completely radical approach to robotics is taken by Ishiguro (2005), who aims at establishing Android Science, by building robots that not only act but also look like humans too. The use of life-size and life-like android robots to improve the quality and naturalness of communication is, however, somewhat controversial, and emprical results of the subjects' perception seem to confirm the "Uncanny Valley" hypothesis. The hypothesis says that the human users first feel increasingly positive and emphatic towards humanoid robots, but at some point when the robot's appearance and behaviour becomes "almost human", the emotional response becomes negative. If the development towards more human-like robots continues, the emotional response becomes positive again.

Software agents can be classified into different types depending on their capabilities and internal processing. For instance, Russel and Norwig (2003) define the classes of agents as follows: the reflex agents respond immediately to environmental percepts, the goal-based agents act so as to achieve particular goal(s) and utility-based agents try to maximise their own "happiness".

Yet the third use of the agent metaphor is found in software engineering: that of software agents. Software agents are computational means to design and

implement software, and should not be confused with ECAs or BDI-agents. They are not autonomous or intelligent agents but software constructs which "live" in system architectures. Originally as a complementary metaphor to direct manipulation interfaces (cf. discussion in Maes & Shneiderman, 1997), they can be used to automatize some tasks and services for the user, by learning to organise and filter information according to the user's personalised needs (thus they help the user to cope with the increased amount of information and maintain a close connection to interface agents discussed above). The agents can also interact with each other and their interaction can be described using the same metaphor as communication, i.e. "speech acts", defining the pieces of information being exchanged (cf. the FIPA Agent Communication specifications). Software agents can also be classified into different types depending on their capabilities and internal processing. For instance, Russel and Norwig (1995) define the agent widely as anything that can perceive its environment and act upon that environment, and then consider the types of programs that the agents can implement to map their percepts to actions, as follows: the reflex agents respond immediately to environmental percepts, the goal-based agents act so as to achieve particular goal(s), and utility-based agents try to maximize their own "happiness".

The agent-based system architectures usually refer to object-oriented programming frameworks within which the application design is implemented. In this case the software agents are pieces of software that encode specific methods to accomplish a task, and they can be grouped into software libraries that will also be available for other applications. The benefits of software agents concern distributed and asynchronous processing, which provide freedom from a tight pipeline processing and can speed up the system performance as well as support a more flexible functioning of a complex system. The need for decomposing the main task into smaller indpendent subtasks usually also helps to resolve the main task more easily than if done in a single block computation. A distributed architecture can also allow experimentation with different computational models of the task and consequently, support flexibility on the level of system design and architecture. However, the design of practical applications using software agents is a skill requiring specialised knowledge: some reference architectures exist (see e.g. Jokinen and McTear, forthcoming), but on the concrete level of designing the system, the task and user requirements are varied and depend on the particular task, domain and interaction management goals.

Progress in various research areas related to building intelligent interactive systems has been remarkable in recent years, and both engineering and human communication studies have yielded important insights in interaction and context management. Developments in speech recognition and synthesis, natural language processing, computational methods and techniques, architectures and hardware design have made it possible to build interactive systems that can be applied in real-world tasks. Despite the advances in technology, the natural interaction mode in various interactive applications is mouse, menu and graphics, rather than natural

language. However, dialogue capabilities allow the system to function in a wider range of situations, and provide services in an easier and more natural manner.

One of the early advocates of intelligent dialogue interfaces was Negroponte (1970) who, argued in favour of a rich interaction language for designer-machine interaction. The starting point was to remove barriers between architects and computing machines, and to enable computer-aided design to have machine intelligence that would allow cooperative, natural interaction. A rich interaction language would bridge the gap between a designer who wants to do something and the execution of the design: the designer need not be a specialist "knobs and dials" person but could formulate the problem and specify goals in his or her idiomatic language, while the design machine could respond, in the same language, concerning the constraints and possibilities of the design.

Most recently, Sadek (1999) pointed out that it is the system's intelligence that provides the good interface and ergonomy of the service. He argued in favour of rational dialogue agents, i.e. a design paradigm for practical systems based on formal modelling of rational agents and their reasoning. Machine intelligence, as the fundamental goal for natural interaction, is possible via a flexible framework that allows rationality, communication and cooperation to be taken into account in the functional requirements of the system, and supports reasoning about mental attitudes and communicative acts.

However, creating a fully-fledged conversational agent is a highly demanding task. As argued in Jokinen (2000) dialogue systems should be investigated as learning systems rather than static models of service providers: the intelligence of a dialogue system depends on the available interaction patterns as well as on the factual and conversational knowledge that the system can deploy in its rational reasoning. In order to equip dialogue systems with the appropriate knowledge, it is not sufficient, or even possible to store the world knowledge explicitly in the computer memory (although with all the information in the web and digital libraries this now seems more likely than a couple of years ago), but it is necessary to include reasoning and dynamic reasoning and update procedures in the system so as to enable the system to use its knowledge and to learn and adapt itself to new environments and to new interaction patterns. For intelligent interaction, it is thus necessary to study how the agents use their full conversational possibilities, including verbal and non-verbal interaction strategies, recognition of new and old information and information presentation via different modalities.

1.2 Design Models for Interactive Systems

The system design needs to address the following questions:

1. What should the system do?
2. How should the system do what it is meant to do?

The first question leads to the design of system functionality, including the design of the system's interaction model. The second question is directed towards architectural and processing issues: when the models of the system functionality, the task and the required interaction capabilities are available, the next step is to decide how to implement the models.

In this section we discuss solutions to the second question, loosely following the agent classification presented by Russel and Norwig (2003), see Section 1.1.2, and focusing especially on the consequences the different types of agents have on the interaction management between the user and the system.

1.2.1 Reactive Interaction

In the simplest case, human-computer interaction can be enabled by the direct mapping of the user input onto a certain system reaction. Interaction is thus managed by specifying the relation between input-output pairs, and reduced to a automatic reaction to a particular input. Figure 1.1 below represents this kind of situation.

Reactive systems are good for tasks where the desired actions are not too numerous to list and the input conditions can be clearly defined. Typical computer interactions are of this type: by giving a command, the user will enable a certain task to be done, and as an answer to the command receive requested information or a message confirming that the requested action has been taken. Many practical applications and speech-based interactive systems are also based on this type of interaction: for instance, banking cash machines and traffic information providers, robot vacuum cleaners and movement detectors. In reactive systems, interaction management is included in the interface design, and there is no explicit model of the interaction: this is implicit in the system functionality. Interaction is managed with the help of scripts which describe the various input possibilities open for the user, and the outputs that the system should provide as a reaction to each input. This can be implemented as a finite-state machine, with deterministic state transitions defining input conditions and output reaction. The rationality of the interaction is based on the designer's understanding of what kind of functionality is needed for successful task completion and for the application to operate smoothly, but the system itself cannot reason e.g. what the inputs are good for or why the actions are executed in a

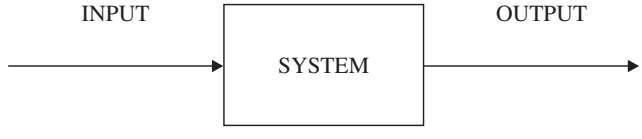

Figure 1.1 Straightforward reaction

certain order. On the other hand, in many cases reactive systems provide a reliable and sufficient way of managing simple interactions, and in some cases, like taking care of frequently occurring dialogue phenomena, they can speed up the system performance, and also provide some analogue for human reactive behaviour.

1.2.2 Interaction Modelling

If the task becomes more complicated and requires an elaborated combination of the input and output conditions, reactive systems will rapidly become cumbersome. The task may also require that some pieces of information must be known before a certain action can be taken, but the manner and order in which this information is obtained is unimportant: the information may be supplied by the user in one go, or the system may acquire it piece-meal through internal actions. For instance, in travel service systems, a user may have asked for travel information by giving information about the departure and arrival locations but no departure time ("I want to go from Eira to Pasila"), while another user may ask a similar question by giving information about the departure location and departure time but no arrival location ("I want to leave from Eira at about 11 am"). Although it is possible to identify the alternatives under different input conditions, this solution fails to make the generalisation that in both cases the user has given information about the departure location, while other necessary information is still to be found out. Moreover, when the amount of missing information increases, possible parameter combinations also increase, and the clear structure of input-output relations starts to fall apart.

In order to manage this kind of interaction in a flexible way, we need to separate the knowledge about the task from the knowledge about the interaction. Although interaction and task management are closely intertwined, they are conceptually separate models: the task model describes the agent's knowledge of what is needed to complete a particular task, while the interaction model describes procedural knowledge of how to achieve the goal through interacting with the partner. For instance, in order to complete the task of retrieving travel information, it is necessary to know the departure place, the arrival place and the departure or arrival time, and if any of the necessary information is missing, the agent needs to formulate a query, and enter into interaction with the user on that particular point. Task model is often simply a list of task-related concepts that the system needs to talk about, although hierarchical task structures and elementary reasoning on subtasks and possible alternatives can be included (cf. expert systems in the 1980's and recent developments in the semantic web). The interaction itself can be understood in different ways, and has been described as straightforward question-answering, negotiation, information updating, plan execution, collaboration on a particular task, decision making, or, as in this book, construction of a shared context in which the agents act. Figure 1.2 depicts this kind of systems

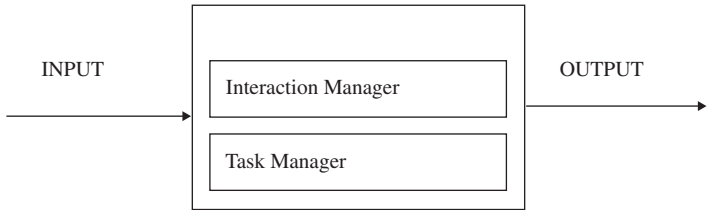

Figure 1.2 Interaction management

with two separate components (managers) taking care of the processing of the two types of knowledge.

In dialogue management, this kind of interaction can be enabled by various techniques where the system maintains a model of the task in the form of a plan or a frame, and operates on the knowledge either in a procedural way as part of the implementation of the whole system, or in a more transparent way by maintaining a separate interaction model where possible dialogue acts, moves and updates are described in a declarative way (see different modelling approaches in Section 2.2 and dialogue management issues in Section 2.3).

1.2.3 Communication Modelling

Although the system's flexibility can be improved with a separate dialogue model component, the computer agents are assumed to operate in a world that is accessible and deterministic: the knowledge is complete for the purposes of the task that the agents want to perform. The designer of the system is an omniscient creator of a virtual world, where the two participants, i.e. the application and the user, interact under logically closed knowledge conditions. The interactive situation can be depicted as in Figure 1.3.

However, the closed-world approach does not lend itself to situations where the partners' knowledge is too large and varied to be enumerated by the designer in advance. For instance, in various ubiquitous computing situations it may be impossible to know and manage all the information or even all the relevant information for a given task, since the agents and the world constantly change. The knowledge is also inherently limited to the particular view-point of the agent at a given moment and the agents' deliberation of the suitable actions is based on heuristics rather than logical inference (rational agents exercise "minimal rationality", cf. Cherniak, 1986). The agents do not necessarily act so as to follow the partner's wishes benevolently either; quite the contrary, the agents can have conflicting goals, and they need to negotiate about the alternative goals. Moreover, interaction does not only take place between two partners but can include a group where communication follows group dynamics rather than individual dialogue strategies.

The interaction between agents in constraint rationality conditions resembles the situation in Figure 1.4. The two agents have their own knowledge of the world

Figure 1.3 Omniscient view of dialogue management

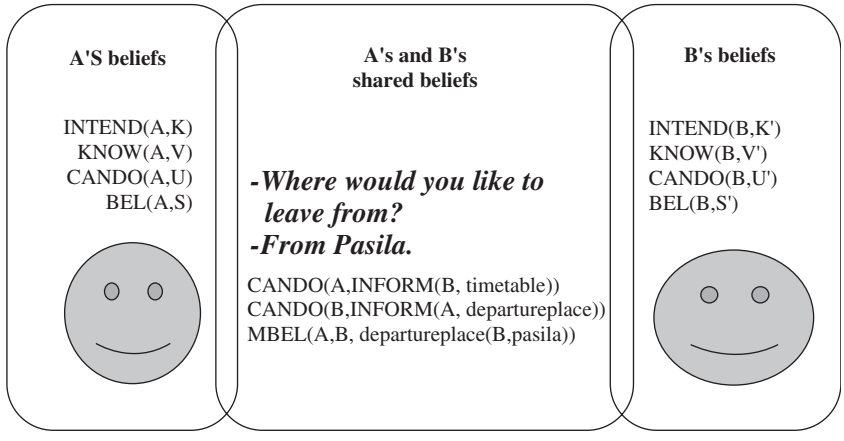

Figure 1.4 Constraint Rationality

and the communicative situation, as well as their own intentions and motivation for the dialogue. Their actions are based on their deliberation with respect to their private beliefs and intentions, and what has been gathered during the interaction as mutual beliefs. Mutual beliefs are part of the shared knowledge, constructed during the dialogue through the presentation and acceptance cycle (Clark, 1989). The model does not assume an omniscient designer but rather, the necessary world is constructed in the course of the interaction.

Traditionally dialogue management models follow the logical structure of the interaction management task: input analysis, dialogue handling, output generation

(see Bernsen et al., 1998). This supports pipelined architectures where the system components operate in a fixed order, and the dialogue manager is seen as one single module in the overall cycle of interaction. More adaptable dialogue management models became available when object-oriented programming started to gain popularity. Agent-based architectures provide flexibility and asynchronous processing so that the components can, in principle, operate independently yet in an integrated manner on the dialogue data. Of course, the logical order of the management task must also be respected, so in practise, the components tend to work in a particular order. However, the order is not meant to be imposed by chaining the software agents to follow the interaction cycle but by making the agents react to follow a particular occurrence pattern in the information that is available in the system: the agents can react to particular information states only, while their reactions can change the current state of system so that it becomes suitable for another agent to react. The order is thus a side-effect of the available information at a particular time in the system as a whole.

The script- and frame-based dialogue management models are not well suited to take care of the communication between agents that interact in constraint rationality situations. In these situations, dialogue management requires special modules for the agents' intention and belief reasoning, as well as efficient coordination of various modalities and multiple knowledge sources, and the basic versions of these models do not offer rich and flexible enough tools for this. Communication between agents in constraint rationality situations requires special modules for the agents' intention and belief reasoning, as well as efficient coordination of various modalities and multiple knowledge sources. Consequently, dialogue management requires rich and flexible enough tools for this. Traditionally dialogue management has followed the logical structure of the interaction management task: input analysis, dialogue handling, and output generation (see Bernsen et al., 1998). This supports pipelined architectures where the system components operate in a fixed order, and the dialogue manager is one single module in the overall architecture.

More adaptable dialogue management became available when object-oriented programming started to gain popularity. Agent-based architectures provide flexibility and asynchronous processing, so that the components can, in principle, operate independently yet in an integrated manner on the dialogue data (see some discussion e.g. in Blaylock, Allen & Ferguson, 2002; Kerminen & Jokinen, 2003). Of course, the logical order of the management task must also be respected, so in practise the components tend to work in a particular order. However, the order is not meant to be imposed by chaining the software agents to follow the interaction cycle but by making the agents react to a particular occurence pattern in the information that is available in the system: the agents can react to particular information states only, while their reactions can change the current state of system so that it becomes suitable for another agent to react. The order is thus a side effect of the available information at a particular time in the system as a whole.

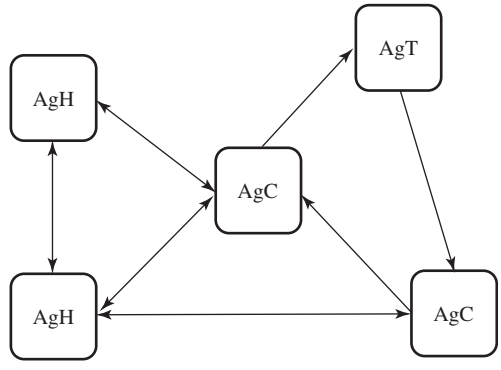

Figure 1.5 Schematic depiction of the interface in Agent-based dialogue management

Agent-based dialogue management thus seems to offer a framework for responsive and constructive dialogue management, especially in the context of ubiquitous communication environment where an increasing number of interactive intelligent applications occur and require the users' attention. Figure 1.5. gives a schematic depiction of the interface in this kind of interaction: it shows the communication modelling approach with different actors interacting, cooperating and coordinating their actions with each other (some of the participants are humans (AgH), some intelligent computer agents (AgC), while others (AgT) do not necessarily have any characteristics of agency or constraint rationality, but are controlled by humans and the more autonomous agents).

1.3 Human Aspects in Dialogue System Design

The emphasis in this book is on natural communication, which is regarded as the main source for intelligent human-computer interaction. There are also many other disciplines that deal with the human aspects of interaction modelling and, the main approaches are discussed briefly in this section. The different departure points cause variations in the primary research topics and methods, but they all share the main goal of designing usable systems and useful artefacts for practical tasks. The discussion leads to the conclusion of this chapter, namely that in designing interactive systems that address the requirements for intelligent interaction within the ubiquitous computing paradigm, the most important human factor is, in fact, the capability for natural intuitive and cooperative communication.

1.3.1 User-centred Design

User-centred design aims at designing end-products that are usable in the specified context of use. The approach emphasises the role of the user in the design cycle, and derives its name from the involvement of the users in the design and testing of the prototypes. The users' are taken into account early in the design cycle

in order to provide information on the desired features and influence the system development in the early stages. Evaluation typically focuses on the quality of interaction between the users and the products, i.e. on usability. The product development proceeds through iterative refinements whereby the users' feedback is used to modify and further elaborate the initial design (Norman & Draper, 1986; Daly-Jones et al., 1999). The users' tasks and the environment need to be understood, too, thus bringing into focus a wide context of organisational, technical and physical factors. Computer applications are seen as end-products which should be effective, efficient and satisfying to use. Special attention is also paid to the understanding and specification of the context of use. The application should be designed with reference to the characteristics of the intended users such as their knowledge, skill, experience, habits and motor-sensory capabilities. The usability of the product can be measured with heuristics that describe the fitness of the product with respect to a set of heuristic parameters (Norman, 1988; Nielsen, 1994).

1.3.2 Ergonomics

Ergonomics looks at the design process from the point of view of human anatomy, physiology and psychology. Its goal is to apply scientific information to the design of objects, systems and environment so that the user's capabilities and limitations are taken into account. For instance, Wickens and Holland (2000) discuss how human perception and cognitive processing works, and also provide various examples of how cognitive psychology can be taken into account in designing automated systems.

In human-computer interaction, much research has focused on ergonomy. Starting from the design of the computer hardware (keyboard, screen, mouse, etc.), ergonomics is also applied to interface and software design so that they would support easy manipulation and transparent functionality of the system.

An important concept in this respect is the cognitive load. It refers to the demands that are placed on a person's memory by the task they are performing and by the situation they find themselves in. It depends on the type of sensory information, amount of information that needs to be remembered, time and communication limits, language and other simultaneous thinking processes. Several investigations have been conducted on automatically detecting the symptoms of cognitive load and interpreting them with respect to the user's behaviour (e.g. Berthold & Jameson, 1999: Mueller et al., 2001). However, it is difficult to operationalise cognitive load and provide guidelines for its measurement. It is obvious that cognitive load increases if the user's attention needs to be split between several information sources, but the different types of information, e.g. visual and spoken, can also effectively support each other. For instance, when studying different properties of presentation styles in a mobile device, Kray et al. (2003) noticed that cognitive load for textual and spoken presentation is low, but when

the more complicated visual information is used, or more complicated skills are needed to master the interface, cognitive load increases rapidly and the interface becomes uncomfortable.

The work on interface design has focused especially on system prompts which should elicit required user input successfully as an answer to the system question. Best-practice design principles emphasise clear, unambiguous prompts which provide the user with explicit information about the task and the system's capabilities. For instance, the 20 Laws of Interface Design, proposed by Weinschenk and Barker (2000) introduce such aspects as linguistic clarity, simplicity, predictability, accuracy, suitable tempo, consistency, precision, forgiveness and responsiveness, in order to make the interface easy and transparent to use. Another important feature of the interface is to help the user to feel in control of the application: it is a tool that the users use in order to accomplish a certain task, and to encourage positive experience, the users should feel that they master the usage of the tool (Shneiderman, 1998). Yankelovich (1996) pointed out that the systems should also prompt the users with appropriate help which guides the users if they are having problems. She suggested different design strategies to cope with different users so that the interaction appears to be helpful. For instance, her own SpeechActs design builds on the requirements of how to design system prompts that show the development of the interaction and address problems with the interaction.

1.3.3 User Modelling

As discussed in Section 1.2.3, the aim of conversational dialogue management is to give means for flexible and rational human-agent interaction by studying human communicative capabilities and building models for their computational treatment. Research has focused on such aspects as the speaker's intentions, speech and language errors, response planning, mutual beliefs, shared information and rational cooperation. Individual users are often taken into account by constructing special user models, explicit representations of the properties of a particular user. Traditionally user models have focussed on the user's beliefs and intentions so as to support knowledge intensive reasoning (McCoy, 1988; Paris, 1988; Chin 1989) but in the practical dialogue systems, they often are simple lists of user preferences (for an overview, see: Kobsa & Wahlster, 1989; Fischer 2001).

User models are usually associated with system adaptation. For example, Jokinen et al. (2002) distinguish static adaptation, which refers to the options that the users can choose from when making decisions on interface aspects such as colour or sound, and dynamic adaptation, which refers to on-line adaptation through interaction, e.g. clustering of users on the basis of their similar navigation choices. Static adaptive features can be listed in personal profile files, but dynamic adaptation requires that the system is capable of observing user behaviour, and can learn from it. For instance, the user's familiarity with the system functionality can be traced on-line, and the system responses adapted to the suitable competence level

(Jokinen, 2005b; Jokinen & Kanto, 2004). A realistic on-line adaptation would further require dynamic updates in the system's knowledge-base, i.e. knowledge to be modelled as clusters of pieces of information that can change according to the interactions with the user.

Concerning computer agents, two opposite approaches have been prevalent in developing user models for them (Fischer, 2001): one with the goal of endowing computers with human-like abilities (human emulation), and the other where the asymmetry between humans and computers is exploited to develop new possibilities for interaction (human complementation). Since human emulation as such has turned out to be a hard task, user modelling has focussed on the complementing approach and sought for more tractable solutions in good design practises and experimental research on the user's interaction strategies. The fundamental asymmetry between humans and computer agents, however, seems to become less clear as the research and development in various kinds of interface agents, robot applications, and industrial interaction systems advances: computer agents are more likely to be assigned properties that relate their functioning to human rational properties such as cooperation, adaptation, learning. Consequently, user modelling becomes similar to intelligent interaction modelling, and the design of dialogue systems can be approached from user simulations. The latter approach has effectively been taken e.g. in Schatzmann et al., (2005), where the goal of the construction of (statistical) user models is to provide a dialogue partner for the dialogue manager during the manager's learning phase of dialogue strategies.

1.3.4 Communication as a Human Factor

The different approaches all concern human factors and aim at building systems that would to do the "right" thing at the "right" time in the "right" way (Fischer, 2001). Because of their historical roots, the approaches have focused on different aspects of the design process, and have resulted in different practical activities. Nevertheless, all of them emphasise the importance of human factors in the design and development of applications, and place demands on the system functionality from the point of view of human needs, constraints, and enablements.

When assessing the usability of final products and interactive applications, the criteria are related to the system's usefulness in the context, its naturalness, adaptation, and ability to engage the user in the communicative situation. These are also typical properties assigned to human cooperative and rational communication, and in the following chapters we will discuss how these aspects affect the user's perception of the system as an intelligent and cooperative dialogue partner, and thus her satisfaction and trust in the system as a useful and enjoyable information providing agent.

In intelligent interface design, the main human factor thus seems to be the system's ability to communicate, especially in such a way that the user (and the system) can pursue their goals effortlessly.

The main contribution of the book is to emphasise natural communication as a starting point for the interaction research. The use of natural language can be regarded as the ultimate human factor in interactive interface design, and include rationality and cooperative principles in the system architecture as part of the standard models for intelligent interaction management. Support for this is found in the ubiquitous computing paradigm where the human user is one member of a large network of communicating agents: the interaction may be of different types and of different levels of competence, but the same conversational principles apply to each agent. In the intelligent environment, where the agents need to coordinate their tasks, desires, intentions, and social activities, communication can easily become abstract and require communicative capabilities that presuppose natural language-based communication between rational and cooperative agents. In fact, it will be argued that the most affordable (Norman, 1988) interactive interface is based on natural cooperative communication.