Artificial Intelligence Review **18:** 117–157, 2002.
© 2002 *Kluwer Academic Publishers. Printed in the Netherlands.*

User Interfaces and Help Systems: From Helplessness to Intelligent Assistance

SYLVAIN DELISLE[1] and BERNARD MOULIN[2]
[1]*Département de mathématiques et d'informatique, Université du Québec à Trois-Rivières, Trois-Rivières, Québec, Canada, G9A 5H7 (E-mail: Sylvain_Delisle@uqtr.ca);*
[2]*Département d'informatique, Université Laval, Ste-Foy, Québec, Canada, G1K 7P4 (E-mail: moulin@ift.ulaval.ca)*

Abstract. Despite a large body of multidisciplinary research on helpful and user-oriented interface design, help facilities found in most commercial software are so ill-conceived that they are often 'unhelpful'. From a wide spectrum of disciplines and software tools, we present an extensive review of related work, identifying their limitations as well as their most promising aspects. Using this material, we attempt to recapitulate the necessary requirements for useful help systems.

Keywords: human-computer interaction, (intelligent, graphical) user interface, (intelligent) help and user assistance

1. Introduction

Since the introduction of microcomputers in the early eighties, we have witnessed a massive migration of software from main frame environments to the individual-oriented and more flexible personal computers. This movement toward the masses required major adjustments from the software industry. Indeed, software had to be made understandable and easily usable by non-specialist users. In order to support users' needs, the software industry then started to design so-called user-friendly interfaces and to produce manuals that would accompany their software – for an historical review of software tools for the development of user interfaces, see Myers et al. (2000). Such manuals were typically called user manual, user tutorial or user help. Compared with usual technical documentation, such as manual for the programmer or reference manual, user-oriented manuals were quite different in their writing style and contents. Sometimes prepared at great costs by professional writers and pedagogical advisors, sometimes prepared hastily by novice software developers, this documentation was meant to literally 'take the user by the hand' to guide her through the sometimes painful learning process of how to appropriately use the software.

In the nineties, we have seen dramatic changes in how the software industry handled the design of user interfaces and the production of software documentation. User interfaces were dominated by the GUI (graphical user interface) approach based on the direct manipulation paradigm introduced by Shneiderman (1983) – some suggest that the GUI offered an alternative to the romantic vision of natural-language-based user interfaces (Sullivan and Tyler 1991). The main advantage of GUIs was the naturalness of the representation used to relate task domain objects and actions (software functions) to graphical representations that were familiar to the users. Any modern interface had to have icons, buttons, lists, menus, scroll bars, and so on. Today, any software which is not using a GUI looks old-fashioned, almost suspicious. As far as documentation is concerned, hard copies were gradually replaced by electronic 'on-line' versions, hypertext or not, which are now found in almost every professional software. But most of all, high quality, pedagogical user-oriented documentation has almost completely disappeared from software packages. It is also quite remarkable how relatively little space is dedicated to software documentation in software engineering textbooks (e.g., Conger 1994; Pressman 1997; Sommerville 2000): maybe it should not come as a surprise if software engineers and computer scientists show so little interest for software documentation.

As a consequence, users need to buy third party books that explain to them how to use the software they have acquired from software vendors. In today's software, documentation's emphasis is put not so much on explaining how to use the software, but on answering user's questions 'on the fly' and 'contextually'. But this is no easy task, as users have quite diverse requirements, all of which happening in various interaction contexts (Akoumianakis et al. 2000). Software documentation is often very technical or not easily accessible, especially because the vocabulary is often unfamiliar to the user. Moreover, users have difficulties to use software documentation in a timely, and productive manner: this explains the phenomenal commercial success of third-party software documentation.

Several reasons can be invoked to explain this state of affairs; let us simply mention three of them:

- the software industry has become very competitive and one way to cut down on the costs and delivery dates was to reduce documentation to its simplest expression;
- the widespread use of GUIs (graphical user interfaces) led the software industry to believe that user support and documentation is becoming less necessary, thus less important;

- with more and more people becoming computer users, the software industry was tempted to conclude that customers would become increasingly more computer competent and thus would need less assistance.

But this is hardly an acceptable justification: users are usually abandoned to themselves and to their frustrations when using a piece of software, whereas they would need intelligent and co-operative help. Such situations must be remedied for future human-computer interaction to stand any reasonable chance to fulfil people's needs. And blind technological progress will not put an end to this situation! Indeed, as discussed in Myers et al. (1996) and Shneiderman (1999), the pace at which technology changes and at which software companies commercialize new versions of their products, plus the diversity of users for which software is produced, offer strong motivation for making user interfaces much more usable. Myers et al. (1996) note that:

> Although some areas of computer science are maturing and perhaps no longer have the excitement they once did, the current generally felt concern with developing human-centered systems, that is, those that more effectively support people in accomplishing their tasks, is bringing HCI to the center of computer science. We have never had more interest, positive publicity, and recognition of the importance of the area.

In the following, we will put forth some hypotheses as to why current help systems are still inadequate. We will examine in Section 2 some of the different aspects that have to be taken into account in the design of help facilities and discuss in which regards their integration into help systems is still problematic. Next, through the review of new investigation fields such as agent technology (Section 3) and more recent trends such as multimedia interfaces, affective computing and virtual reality (Section 4), we will try to underline those elements of these technologies that are relevant to help systems and can enhance their usability. This will be followed by a general discussion (Section 5) and a conclusion (Section 6).

Details about today's run of the mill help systems or documentation can be found in some software engineering or user interface textbooks and will not be discussed here. All references are listed at the end. We also present a brief list of URLs that will allow the interested reader to find additional information on the World Wide Web. We also wish to remark that several subsections contain non exclusive material: some works could be included in more than one subsection.

2. Main Requirements for Help Systems

Mathews et al. (2000) mention two criteria that need to be discussed when designing assistance systems: (1) the degree to which the system facilitates the accomplishment of a particular task by a user who does not currently know how to do it; (2) the effectiveness of the system in aiding users' learning processes by providing mechanisms that enable them to improve their level of performance as they are involved in their day-to-day activities on the system. Based on these criteria and on their experience in the development of assistance systems, Mathews et al. (2000) suggest that the development of effective assistance systems should be based on a framework that includes: (1) an *ideal model* that is used to explain system concepts and tasks to users; (2) a *user model* which keeps track of what the user is doing, what she knows and does not know; (3) a *natural language interface* which can understand users' queries posed in natural language so that users are not constrained by system terminology and rigid query formats; and (4) a *response generation mechanism* that checks the user proficiencies and the context of user's activities before answering a user query or prompting the user. However, we must observe that the vast majority of help systems available in commercial software do not comply with such requirements.

In order to help a person, one has to first gather relevant information about her knowledge and beliefs about the domain, her goals and the task she is trying to perform to achieve that goal. These requirements for efficient assistance, although easily satisfied by humans, continue to seriously challenge help or explanation system designers. A useful help system must indeed be endowed with knowledge about the user, be able to detect her goals and cooperate with her so that she can successfully perform her task. These issues are dealt with under topics such as user modelling, plan recognition, task modelling, etc. and continue to be at the heart of user interface design. In the following sections, we will review some of these concepts and see to what extent their shortcomings can account for the fact that current help systems are still highly inadequate. An inadequacy that we can already explain by several hypotheses:

- In order to effectively assist a user, a help system should have some understanding of the user's plans and goals, as well as of the task being performed. None of the mainstream software systems provide help facilities based on plan recognition, user modelling or task modelling. However, several research works have addressed the issue of developing various models in order to support the collaboration between users and software systems (see Section 2.1.1).

- Another problem is that users often have misconceptions about the system. They unconsciously construct a faulty and incomplete model of the software with which they interact. The help system must be able to correct the user's model of the system. We will discuss these issues in Section 2.1.2.
- The help system should try to establish a cooperative setting by providing relevant explanations on different aspects of the system when needed, by giving feedback on its proper state and by volunteering information on a proactive mode – more on this in Section 2.1.3.
- The user's competence in the domain of application for which the software was developed has significant impact on her capabilities to efficiently exploit the software's potential. This will be even truer if the user interface has been well designed in order to appropriately reflect the user's perspective when performing tasks in that domain. Similarly, the user's experience with the software has great impact on the type of interaction she will have with the software. A novice user will normally tend to use a very restricted subset of the software's functionality and will also expect more extensive and more tutorial-like assistance when facing difficulties. An experienced user will have a broader grasp of the software's functionality and will expect less intrusive assistance, although probably at a more advanced level. The help system must therefore create and maintain a model of the user with whom the system is interacting. We will turn to this in Section 2.2.
- The last but not the least problem with help system design seems to be that of methodology. We think that the users' needs with regard to a given system must be identified at the early stages of that system's development. This will not only improve the interface design and the representation of the system's different tasks, but will also determine the conceptual and procedural aspects of those tasks that will later have to be explained through the help facility. We will discuss this issue as well as the problem of software evaluation in Section 2.3.

In all fairness to the numerous researchers who have contributed to the above fields, we must recognize that that their works were not all primarily concerned with help systems. We here present a discussion which could hopefully facilitate the identification of potential links between solution elements that, we believe, offer great potential for the future development of intelligent help systems. In fact, an integrated view of such diverse contributions is certainly a much needed ingredient to the development of an "ultimate" theory of human-computer interaction, of which help systems are a necessary component. But, as pointed out in Sutcliffe (2000), delivering human-computer interaction knowledge to (system and software) designers is often limited by scalability problems. Undoubtedly, this is a very difficult problem. This paper does not offer a solution but, rather, a first step to a more integrated vision of how one may approach the issue.

2.1. *Helping the User with Her Task*

Researchers distinguish two types of help systems: passive and active. A *passive* help system answers a user's requests but has no knowledge about her goals, while an *active* help system continually monitors the user's actions, trying to discover her goals from the performed actions and initiates at times the interaction with the user. With active help systems researchers aimed at simulating the behaviour of human consultants who consider users' knowledge and abilities and the context of their activities when answering their queries. In addition, a consultant usually will help a user's learning process by indicating better and more efficient ways of performing tasks, when required. Matthews et al. (2000) mention that in order to achieve such capabilities online systems need to: (1) model individual users by keeping track of their strengths and deficiencies in order to occasionally make suggestions and help users improve their capabilities; (2) determine the context of users' activities and ensure that responses cater to their individual needs and proficiencies; and (3) provide means so that user queries to the system need to be constrained by system commands and terminology.

In the following sub-sections we examine some of these important issues: user's task modeling and representation; the distinction between misconceptions and misunderstandings and how to achieve a cooperative setting. In Section 2.3 we will discuss the issues related to the introduction of a user model in a help system.

2.1.1. *Task representation and task modeling*

HCI researchers concerned with various aspects of user assistance have been interested by the problem of task representation. As Sullivan and Tyler (1991, p. 3) note, besides the problem of inferring the user's knowledge and abilities, the issue that continues to challenge interface designers is a matter of how well an interface addresses the semantics of its task. Shneiderman (1998) who provides a wide coverage of HCI factors is well known for making a strong case for direct manipulation interfaces, i.e., GUIs, which represent task objects with graphical entities. Shneiderman argues that the design of a software user interface should follow an Object-Action Interface Model (OAIM) to represent appropriately, via its GUI, the user task objects and actions. The basic idea is that task objects should appear as objects in the GUI, thanks to a suitable metaphor, and task steps (which are in fact elements of the user's intention) should appear as actions of a plan at the interface level.

The same principle should be applicable to user documentation: see Chase et al. (1993) and Section 12.3 of Shneiderman (1998). Young (1991) suggests that the user interface could use a graphical depiction of the current plan, including completed and undone steps, in order to keep the user aware of the current state at all times.[1] In this regard, Young distinguishes two types of context: a persistent context, provided by knowledge bases containing domain- and user-specific information, and a transient context, provided by ongoing instantiation of interaction plans which represent the task structure of interaction sequences in advance. The second type of context is at the heart of Akoumianakis et al.'s concerns (2000, p. 391): "Achieving context-sensitive processing requires a mechanism for obtaining a global understanding of the task execution contexts". An interesting issue is the relationships between the user's and the system's task representations (Terwilliger and Polson 1997).

The importance of plans and goals is thoroughly discussed in (Carberry 2001, p. 31):

> Knowing a user's plans and goals can significantly improve the effectiveness of an interactive system. However, recognizing such goals and the user's intended plan for achieving them is not an easy task.

The main approach that has been developed to perform plan recognition is that of plan inference. A system using plan inference must have *a priori* knowledge of the set of goals a user is expected to pursue, along with sequences of actions that allows her to accomplish these goals. Thus, to put it very simply here, the system constantly observes the user's actions and attempts to associate these actions to the ones it knows and which are linked to the goals it also knows. When the system is able to do that, it can infer the goals the user is pursuing. The system can thus offer better support to the user's needs because it has some representation (in terms of plans, goals and actions) of what the user is apparently trying to do. An example of a plan recognition system is that of the Unix Consultant (UC) project: see, amongst others, Mayfield (2000). Another example is the plan recognition system used in the CAUTRA air traffic control system (Mo and Crouzet 1999). Many extensions have also been proposed to basic plan inference, ranging from formal reasoning models, based on argumentation or abduction for instance, to probabilistic reasoning models (for instance, see Virvou (1999) for an intelligent help system based on a theory of human plausible reasoning). The idea of monitoring the user's actions in order to determine her intentions and (sub-)goals has also been used in systems not based on plan recognition, but on a much lower level of modelling which takes into account the states of a computer's mouse and cursor (Agah and Tanie 2000).

Task modelling can help with the problem of goal or plan recognition. Goals refer to states that a user wishes to reach and plans describe steps to achieve the goals. Plan recognition becomes a big challenge especially if the user's goals are ambiguous (Ardissono et al. 1993; van Beek et al. 1993) or based on mistaken beliefs. Several works have been devoted to the detection of user's misconceptions (see Section 2.1.2) and to the development of robust and flexible plan recognition systems (Quilici 1989), as well as to the way a system should react to misconceptions (McCoy 1988). Based on his experience of developing the SINIX system, Hecking (2000) discusses the problem of plan recognition in an active help consultant, using symbolic logic. SINIX answers user's natural language questions about SINIX commands, objects and concepts and is capable of activating itself. At the heart of SINIX is a plan recognition sub-system. Hecking shows how an interval-based logic of time can be used to describe actions, atomic plans, non-atomic plans, action execution, simple plan recognition as well as the recognition of inserted subplans. Hecking also discusses the limits of this formalism. Hegner (2000) discusses the problem of representation of dynamic knowledge and proposes a plan generation mechanism which is used to solve complex dynamic queries in the Yucca help system. Yucca provides users with detailed expert advice on the use of UNIX command language, especially to complex queries whose solution may involve the interconnection of several commands, each with multiple options. Interestingly, this research showed that the form of knowledge representation necessary to support modelling of communication with the user is quite different than that which is appropriate for the details about the behavior of an operating system.

Often, the actions proposed by a help system aim at orienting the user to take advantage of the system's functionalities, with no consideration of the user's real needs when accomplishing her current task. Task models can facilitate the identification of the user's current goals and situation. Most of the systems using task models have been developed for problem solving tasks, mainly expert systems, decision support systems or intelligent tutoring systems. We think that task models should be seriously taken into account for the development of other classes of systems, especially in relation to help functionalities.

We find task models and user models (see Section 2.2) in *explanation facilities* which explain to the user how a problem has been solved and why the system reached certain conclusions, as well as in *intelligent tutoring systems (ITS)* which are instructional systems teaching knowledge or skills to users in specific application domains (Wenger 1987). ITSs attempt to get information about the learner's learning state in order to adapt the instruction strategy to fit her needs. This is usually done on the basis of a problem solving model of the task to be accomplished. Based on such a task model, an ITS attempts to determine which tasks the user currently tries to perform, in order to help her when difficulties arise, for example, by giving some hints (Zhou et al. 1999).

In order to control the software development costs, certain researchers have investigated ways to accelerate the creation process of ITS by providing task-specific authoring environments to develop ITSs for classes of tasks (El-Sheikh and Sticklen 1999). The basic assumption of such an approach is that complex problems can be decomposed into 'basic tasks' (problem solving strategies and knowledge representation templates). Hence, an ITS could be automatically generated by developing a shell that can interact with any task-based expert system and produce an ITS for the domain topic addressed by it.[2] Several approaches can be used to develop task models based on well-known knowledge engineering approaches such as KADS (Schreiber et al. 2000) and the Generic Task model (Chandrasekaran 1986) also used for explanation purposes (Tanner et al. 1993), or cognitive task analysis (Lovett 1998): these works should provide inspiration for the development of task-based help systems.

Let us note that what is being referred to as 'task model', usually understood from the user's viewpoint, can also include the agent's task model if both the user and the software agent are modelled as performers of a collaborative task. Such a model can be thought of as a collection of action plans shared by the agent and the user in which the initiative can be taken by the person or the machine. The Collagen system (Rich and Sidner 1997) is an example of such a system which embodies a set of conventions for collaborative discourse as a mechanism for maintaining the flow and coherence of agent-user interactions.

Many researchers emphasised the role of task knowledge as a vital component of intelligent and helpful interfaces (Marion 1999a,b), but also as a means of evaluating the quality of the user interface. Lecerof (1997) proposes to use task models for assessment purposes, while Mitta and Packebush (1995) suggest to measure the learning rate with which the users successfully complete HCI tasks. Such tools could indeed allow software developers to better judge the quality of their work, from the users' point of view, and thus represent a sensible way by which user interfaces and help systems could be put at the very heart of the software team's attention.[3]

If help systems had access to a useful representation of the user's task, they could offer contextually relevant and beneficial help. However, developing help systems based on the techniques (e.g., plan recognition) that would allow them to do that is a complex and knowledge intensive process which, so far, has been confined to prototype (often, artificial intelligence) systems.

2.1.2. *Misconceptions and misunderstandings*

Another problem the system should deal with is that of the user's model of the system. This model corresponds to the user's understanding of the software's functionality, capabilities and behaviour. A model that is regularly updated as the user discovers new aspects of the software. But this model will normally be incomplete and more or less faulty. Incomplete because users very rarely know everything about a software, including its defects and its reaction to unforeseen data. Faulty because users may not have a perfectly correct model of the software: this model might contain errors that could lead to an inappropriate operation of the software – see Virvou (1999) for an intelligent help system built on a reasoning mechanism that focuses on error diagnosis. It might also be the case that discrepancies between reality and the user's model could be caused by undocumented 'bugs' in the software: the user's model might be correct according to the software's normal (and documented) mode of operation but unexpected software behaviour forces the user to update her model. Whatever their origin, we call such discrepancies misconceptions. We distinguish misconceptions from misunderstandings. A *misconception* is normally a mental representation of something for which the person is sure or at least very confident, i.e., for which she believes she is right or has few doubts as to the validity of her knowledge. When a person discovers that what she thought was correct was in fact wrong, she normally reacts with surprise. A *misunderstanding*, which might be the source of a misconception, refers to the user's reaction when faced with a concept or situation which she has trouble understanding. In that case, the person is conscious that she is faced with something for which she does not have a full grasp or for which her knowledge may not be reliable – one might want to argue that some misconceptions are unconscious misunderstandings.

When using GUIs, users acquire personal conceptual models of the application programs laying behind these interfaces (Miller et al. 1987), models that are initially flawed and incomplete. A situation which leads users to various problems. Miller et al. (1987) argue that the user assistance mechanism should be able to anticipate and solve such problems through the diagnosis and repair of these misconceptions. Earlier similar work appeared in McCoy (1983), Smith (1985), and Dede (1986). McCoy points out that by failing to correct user misconceptions, the system may not only appear to confirm the original misconception, but may cause the user to develop even further misconceptions – the same observation is made in Maybury (1992). Chin's Unix Consultant system (Chin 1998) tries to correct (detect) user misconceptions by denying that what the user mistakenly believes is the case. The problem of recognising and handling misconceptions is also an important issue in intelligent tutoring systems (Mallen 1996) and intelligent learning environments (Lester et al. 1997). How exactly misconceptions relate to knowledge gaps between software systems and the users is an interesting issue to investigate (Winograd and Flores 1986).

An ideal help system must minimise the number of misconceptions and misunderstandings, particularly if they prevent the user from attaining her goals. As with user's intentions and plans, misconceptions are difficult to detect but hold the potential, if identified, to adjust assistance to the user's needs and further improve future interaction by allowing the repair of the user's model of the system. In Fischer (2001, p. 80), this problem is identified as a future research challenge ("dealing with user models containing wrong, outdated, and inadequate information"). We should also emphasise the importance of presenting a meaningful interface to the user which relates naturally to her view of the software's application domain in terms of the metaphor, functions and objects used. Also, the terms and concepts used throughout the interface should be consistent with the user's ontology in order to facilitate communication between the human and the machine by minimising areas of potential confusion and misunderstanding.

2.1.3. *Cooperative setting*

A co-operative setting requires that the system provide feedback to the user. Pérez-Quiñones and Sibert (1996) present a model with five feedback states that must be communicated to the user in order to fulfil the communication expectations of a dialogue. These five states are: *ready*, *processing*, *reporting*, *busy-no-response*, and *busy-delayed-response*. These authors found that when the system failed to properly identify one of these states, it led the user to make a tentative repair which could consume a significant amount of time. This provides an example of collaborative HCI, although at a relatively low level. Frohlich et al. (1994) show the importance of signalling to users when a sequence of interaction turns with the computer is complete, much like in human conversations. Moreover, they recognise the need for an on-line repair facility which would be able to handle the user's troubles and guide her when repairs are necessary – the same observation is made in Shneiderman (1999). Work by Weissert (1996) also showed that error messages could be much more useful when accompanied by recovery actions. The same principles are also important in agent-based software, as reported in Sengers (1999).

Providing explanations on the system's reasoning mechanisms is also important for establishing a cooperative setting (Moulin et al. 2002). Hermann et al. (1998) indicate that in the context of software systems, explanations should have two purposes: to make clear all aspects of the system and to provide the causes or reasons for decisions made by the system. The need for introducing explanation capabilities in software systems was recognised during the construction of the first expert systems in the early eighties. In those systems, explanations were merely reasoning traces used by knowledge engineers in order to check the coherence of the knowledge base. It has long been recognised that explanation knowledge should be distinguished from expert knowledge contained in the reasoning component (Wick and Thompson 1992) and that explanations must be seen as a problem-solving issue by themselves. Yet, the automatic generation of useful and understandable explanations remains a difficult task (see, e.g., Cawsey 1998). It should be emphasised that building an explanation component requires resources, energy and time in addition to those devoted to the development of the expert reasoning component, not to mention the effort to adapt the explanations to the user's profile (user's knowledge, vocabulary, reasoning style, etc.).

Several projects (see Intelligent User Interfaces in the web sites listing at the end and AI Magazine 2001) aim at developing collaborative intelligent user interfaces (IUI). For example, Rich et al. (2001) work on the COLLAGEN project and develop a collaborative IUI which exploits the metaphor of conversation. Several prototype systems have been developed using COLLAGEN in which an agent guides the user through the process of achieving a typical task such as the set up and programming of a video cassette recorder, the operation of a gas turbine, and help provided to people when programming a home thermostat. Interactions between the agent and the user are carried out in English. At the core of COLLAGEN is the *discourse state system* which tracks the beliefs and intentions of all participants in a collaboration and provides a focus-of-attention mechanism for tracking shifts in the task and conversational context. COLLAGEN also use plan recognition mechanisms in order to reduce the amount of communication required to maintain a mutual understanding between the user and the agent of their shared plan in a collaborative setting. These techniques could be used to develop advanced help facilities which are able to interact with the user in a conversational setting. Although it is clear that most software projects are not able to develop such advanced help facilities because of tight budgets and delivery dates, some domain of application may benefit from these techniques such as tutoring systems (Graesser et al. 2001) and character-based presentation systems (André and Rist 2001).

The knowledge about the user can be organized in a user model (Section 2.2). By appropriately exploiting the user model maintained by the system (e.g., Komatsu et al. 1994), a system can display several cooperative attitudes. This would result in volunteering information not explicitly asked for by the user (Sæbø 1988; Liu and Ng 1998), handling unexpected exchanges in dialogues (Jokinen 1996), tailoring answers to the user's knowledge level (Paris 1988), or adjusting a system's output to the user's boredom and cognitive overload (Zukerman and McConacky 1995).

There are several ways to define what a cooperative system is, or could be. Here is an excerpt from Tsui and Azvine (2000, p. 259): "It is important for computers to proactively assist humans, anticipate the effects of their action and learn from users' reactions". Cooperation is definitely a highly desirable feature of help systems. But very few modern systems could qualify as even mildly cooperative. All too often, the user is under the impression she has to fight against the system to either get the required help or accomplish her current goal. Tsui and Azvine argue that one way to improve interaction between humans and computers, and thus to minimize communication ambiguities, is through multimodal user interfaces – see Section 4.1. However, this is only part of the picture. Sooner or later, even in a situation of perfect human-computer communication, the user will require help. And then, the dimensions covered in other sections of this paper offer invaluable elements of solution.

2.2. *Learning about the user*

In order to adapt their interactions to the user's knowledge, style and other characteristics, some systems use and maintain a model of the user. The idea that computers should adapt to their users has been a major concern in both Human Computer Interaction (HCI) and Artificial Intelligence (AI) fields, including Computational Linguistics, for more than fifteen years (McTear 1993; Schneider-Hufschmidt et al. 1993; Kobsa 2001) – for a recent review of user modelling research, see Fischer (2001). In addition, works in cognitive and differential psychology show that user analysis is central to well-designed systems (Dillon and Watson 1996). We mentioned already in Section 2.1 that researchers distinguish two categories of help systems, depending on how the interaction with the user is initiated. A passive system answer users' initiated queries, while an active system initiates the interaction with the user at times, when it identifies an opportunity to help the user. Based on an empirical study of how various kinds of users in an academic site manipulate UNIX files, Virvou et al. (2000) discuss the advantages and limitations of an active help system for UNIX and the type of help needed by users. These authors suggest that in order to provide such help, the construction and maintenance of a model of each user is required. In HCI research on user models, a special emphasis is put on the user's cognitive style and personality factors. AI approaches of user modelling mainly aimed at developing systems that can automatically construct a model of a user as she interacts with the system. The user's characteristics that are typically modelled are: goals and plans, capabilities, attitudes and preferences, knowledge and beliefs. The same features are modelled in current software agents models (Moulin and Chaib-draa 1996; Singh et al. 1999).

How is the user's information acquired? McTear (1993) identifies the explicit and implicit modes of acquisition. In the *explicit information acquisition* mode, knowledge about the user is obtained through user-system interactions (Linden et al. 1997), while in the *implicit information acquisition* mode, it is acquired by inference mechanisms, or even by mechanisms based on neural networks (Ye 1997). This knowledge may be relative to an individual user or to a *stereotype* whose characteristics are shared by several users (Chin 1989; Finin 1989; Rich 1998). Of course, contradictions may appear if a user is assigned to the wrong stereotype or if the stereotype's default values do not apply to a given user. Several other AI techniques have been tested to enhance user models such as machine learning techniques, neural networks and Bayesian networks. We will not discuss those techniques here, but the interested reader may consult the website of the *Association of Researchers Working on User Modelling* (http://www.um.org/).

User models can be used to tailor the system's responses to the user's needs and to interpret the user's input; hence, their relevance to system-generated explanations. Explanations will be better understood if they are adapted to the user's knowledge of the domain and if they take into account the user's goals, plans and preferences. The information contained in a user model can even be updated through an explanatory dialogue (Cawsey 1998).

User modelling has also been addressed in research on intelligent tutoring systems (ITS) with the creation of student models representing a student's knowledge and understanding of the material being taught (Lelouche 1998). Various researchers proposed to characterize a student's knowledge with respect to the system's knowledge of the domain. However, such an approach cannot be used to characterize a student's incorrect knowledge. For that purpose, a system might contain 'bug catalogues' or 'mal-rules' in which are recorded typical students' errors for a given topic. At a more sophisticated level, student modelling may involve plan recognition: a system tries to find out why a student asked a particular question or performed some action in order to relate these elements to its beliefs about the student's goals. Such a system should also be able to identify the cases in which a student lacks knowledge or has misconceptions about what is being taught.

Naturally, user models have been used in intelligent help systems. Among other works, let us mention Cesta and Romano's (1989) help system that takes into account the user's state of knowledge, intentions and plans, as well as her communicative abilities and Chiu et al.'s (1993) adaptive help system that can operate with various kinds of users, from novice to experienced. Jones et al. (2000) developed an active intelligent help system that monitors users in order to offer spontaneous help when they are facing problems while using UNIX. The system builds and maintains a model of the user. The user may have misconceptions about how commands work or about what is true in the current system state. The authors describe a mechanism of the user modelling component which accounts for different hypotheses about what the user is actually thinking at every stage of the interaction. The consistency of these assumptions if managed by an Assumption-based Truth Maintenance System (ATMS). The selection between a number of different user models is based on information which is extracted from the feedback that UNIX provides in response to user actions.

We must mention here the scepticism of certain researchers with regard to the feasibility of building true user models. From a cognitive standpoint, the adequacy of approaches to user modelling are evaluated according to the degree to which they are able to faithfully represent relevant aspects of the users. Gilbert (1987) argued that even the most sophisticated approaches fell short of that goal. The reason is that these approaches are based on modelling the assumed cognitive states of users, including their plans, intentions and goals. Drawing on studies of human-human interaction, Gilbert argued that the cognitive states of other interactants are not available to a speaker although adaptation to their conversational moves appears to be successfully achieved. Gilbert concludes that attempts to construct better models of the user's cognitive state are misconceived and that the emphasis should be put on providing software with models of the system itself and models of the interaction, in order to allow 'reflexive reasoning' and 'meta-level' commentary on the user/system dialogue (Gilbert 1987). Ramscar et al. (1997) also emphasise the weakness of the current 'knowledge modelling' due to the absence of any convincing psychological model of conceptual categorisation. This issue is important since systems try to categorize users according to stereotypes. Ramscar et al. (1997) argue that a system's representations need not be definitive, but rather must function 'pragmatically'. The system should be able to adapt its user model when an external representation of the user appears to be 'aligned' structurally with some of its stored representations: hence, it could pragmatically attribute certain conceptual knowledge to the user.

On that issue it is worthwhile mentioning Strachan et al.'s (1997, 2000) experience of building a 'pragmatic user model' for a commercial software system: TIMS, the *Tax and Investment Management Strategizer*. They state that in spite of the demonstrated capabilities of many research systems including user models, little progress has been made in commercial software systems. They suggest several reasons for this situation: (1) the performance overhead on the system when including a user model; (2) the time and expenses involved in the development of a user model component; (3) the fact that many approaches embodied in research systems are too complex or impracticable for use in commercial software systems. This situation has led researchers to create 'pragmatic' user modelling techniques based on theoretically-motivated approaches. From a commercial standpoint, this amounts to include in software systems 'a minimalist user model component' benefiting from the main advantages of large research systems, with minimal cost and commercial disruption. The authors present the main issues related to the design, implementation and empirical evaluation of such a minimalist user model targeted to the novice users of the TIMS system. They raise interesting issues such as the importance of measuring the practical impact on users of the introduction of a user model component in a software. They also emphasise that users should be involved early in the system development process so that their 'real needs' with respect to their use of the software under development can be identified. Chin (2001) also emphasise the need to carry out empirical evaluations (i.e., appraisal of a theory by observation in experiments) in order to determine if users are helped of hindered by user-adapted interaction in user modeling systems. These issues are discussed in the next section.

2.3. *Help systems: methodology and evaluation*

We are conscious that in current software development practice the effort devoted to the creation of a help facility is only a small part of the effort assigned to the overall system development, and such a situation will not change in the foreseeable future. However, we are convinced that something has to be done in order to improve the quality of software when it comes to the help that users should receive from it. We think that the software development process should be adapted in order to take into account users' requirements and needs for the help component of the future system. The help component should appear in the early versions of the software architecture so that designers keep in mind that it is an important part of the system. When gathering users' requirements, designers should also ask questions in order to obtain information about their vocabulary and the main tasks they will perform using the future software. This information is much needed in order to build usable interfaces and to take into account the context in which the system will be used, hence, a better starting point in order to build the help facility. The help component should be built in parallel with the software under development because it must be thought of as tightly coupled with the system interface. As soon as the design team has decided upon the main system functionalities and the main system interface characteristics, the reflection about the help component can start, in order to make sure that the help functions will take into account the context in which the user will use the system. The first versions of the help system should be delivered early, at an appropriate date during the overall system construction, but certainly much sooner than is usually done in current system development (i.e., after most of the system is delivered).

Certain readers may argue that it is often the case that designers develop a new system without having proper users' requirements because they are developing an innovative system and have no access to potential users. This happens when developing new software that is expected to fulfil new needs not yet addressed by the competition. However, if the software is to be sold, it must in any case be useful to future users. Project managers controlling the software development should also be convinced of the importance of designing systems for usability and hence integrate in software development plans and schedules the required resources for help facilities.

Some of these issues have been taken into consideration in user-centred design and usability engineering. We think that the same principles should apply to the design of help facilities. User-centred design methods (Norman and Draper 1986; Preece 1994) emphasise the fact that design should: (1) be user-centred and involve the users as much as possible so that they can influence the design; (2) integrate knowledge and expertise from different disciplines that contribute to HCI design, and (3) be highly interactive so that testing can be done to check that the design does meet users' requirements.

Numerous papers and books have been written on the subject since the idea of design for usability has been introduced by Gould and Lewis (1984), some of which explore how cognitive science can be used for user interface design (Gardiner and Christie 1987; Norman 1984).

From these seminal works emerged the modern discipline of Usability Engineering which aims at supporting the entire software development process with user-centred design and validation activities.[4] Of interest are also the works done in the field of Cognitive Ergonomics at the intersection of Psychology and Design.[5] Finally, we must mention that with the explosion of internet applications (e-commerce and the like) usability design is currently attracting much interest for the development of user-centred Web applications.[6]

In Akoumianakis et al. (2000, p. 391), the authors propose what they call the unified interface development method:

> A unified user interface is defined as a hierarchical construction in which the root represents an abstract design pattern de-coupled from the specifics of the target user (group) profile and the underlying interface development toolkit, while leafs depict concrete physical instantiations of the abstract design pattern. The unified user interface development

> method comprises design- and implementation-oriented techniques for accomplishing specific objectives.

The first and fourth specific objectives of their user interface design are (*ibid*, p. 392):

> "accommodation of different user groups with varying abilities, skills, requirements and preferences, as opposed to the average 'able' bodied user", and "propagation of design knowledge into the development cycle by embedding design recommendations into user interface implementation".

These authors did not take into consideration the development of the help facility as such. However, the methodology they propose could easily be extended to include the help facility.

Another important problem is the evaluation of the user interface. Based on work by Senach (1990), Balbo (1994) investigated the automation of user-interface evaluation. Balbo considers that the evaluation of software ergonomics should be based on two main dimensions: utility and usability.[7] She also argues that software design should be centred on the user in order to properly take into account ergonomic considerations. To this end, tasks should be described from the user's viewpoint, where a task is defined as the combination of a goal plus a procedure. Balbo presents a tool called ÉMA (Évaluation par Mécanisme Automatique) which automatically produces ergonomic evaluations of specific user-software situation. ÉMA uses a rule base of user behaviour models, a forma representation of potential tasks for the specific software at hand, and behaviour data recorded from the user during her interaction with the software. User behaviour data consists of a file of events that took place during the evaluation. Potential tasks are represented using so-called sequence graphs, which are directed graphs organized in terms of a hierarchy of procedures expressed with certain constraints (sequence, obligation, restriction). During a particular evaluation, both sequence graphs and files of events are loaded into ÉMA's analyzer which then produces annotated versions of the two inputs. This analysis, based on ergonomic principles developed by Scapin (1990) (i.e., guidance, work load, explicit control, adaptability, error management, homogeneity, codes significance, and compatibility), detects anomalies which are then reported and expected to be further analysed by a human expert who will eventually adapt the software to better meet the users' needs. Even though ÉMA is described as an automatic tool, it is more like a semi-automatic tool, since human expertise is required first to define the models needed by the analyzer and second to interpret the analyser's output. But nevertheless, it seems to be a good idea to monitor the user's behaviour, with dedicated tools, in order to better adjust the software to her needs, instead of always assuming that the user will adjust herself to the software. Clearly, a good evaluation of the user interface will provide a useful feedback to the team responsible for the development of the help facility, allowing them to put emphasis in those areas where appropriate help is need to better support the user.

In the following sections we investigate new fields such as agent technology (Section 3), multimedia interfaces, affective computing and virtual reality (Section 4) and underline certain elements of the corresponding technologies which may be relevant to help systems and can enhance their usability.

3. Agent Technology

Agent technology is more and more used for assistance purposes. In this section we will try to see how these intelligent agents can fulfil the users' assistance needs. But, first we have to define the notion of an agent. There is still no consensus on what exactly an agent is: several definitions are proposed in Bradshaw (1997), Russell and Norvig (1995), and Hayes-Roth (1995). Weiss proposes the following definition (1999, p. 27):

> . . . for an increasingly large number of applications, we require systems that can decide for themselves what they need to do in order to satisfy their design objectives. Such computer systems are known as *agents*. Agents that must operate robustly in rapidly changing, unpredictable, or open environments, where there is a significant possibility that actions can *fail* are known as *intelligent agents*, or sometimes *autonomous agents*.

For a recent discussion of agent-based software, see Jennings (1999); for a discussion on the notion of autonomy, see Friedman and Nissenbaum (1997). Finally, for a critical analysis of agents, see Shneiderman (1997).

In Sections 3.1 and 3.2, we discuss two types of agents: interface agents which automatically perform certain actions for the user and intelligent assistants or intelligent support systems which support the user in her task.

3.1. *Interface agents*

Interface agents are programs that can affect the objects in a direct manipulation interface without explicit instruction from the user.

Maes and Kozierok (1993) define interface agents as computer programs that employ artificial intelligence techniques in order to provide assistance to a user dealing with a particular computer application. Modelled closely after the metaphor of a *personal assistant*, the agent discussed in their paper learns how to assist the user by (i) observing the user's actions and imitating them, (ii) receiving user feedback when it takes wrong actions and (iii) being trained by the user on the basis of hypothetical examples.

The basic idea presented by the authors is that the interface agent learns by "continuously looking over the shoulder of the user as the user is performing actions", and this over long periods of time. For instance, if the agent finds that action X is often performed after action Y, then it will suggest to automatically perform X whenever Y is performed. Such interface agents are often qualified as semi-intelligent and semi-autonomous. But there are serious limitations with these agents. They assume that the user knows what she is doing and they can only learn after a certain number of repetitions or in situations which are very similar to previous ones – see Lashkari et al. (1994), Takeda et al. (1996), Decker et al. (1997), and Good et al. (1999) for frameworks for multi-agent collaboration which allow greater flexibility in learning and information exchange.

Lieberman (1997) argues that agents should be both interface agents and autonomous agents. Such an agent is a program operating in parallel with the user and does not follow the traditional conversational approach in which the user and the agent alternate acting or follow a collaborative dialog (Rich and Sidner 1997). LETIZIA is the name of Lieberman's system, an autonomous interface agent for Web browsing. LETIZIA is constantly searching the Web, according to preferences and interests specified by the user, and presenting its results (candidate sites and recommended sites) to the user in dedicated Netscape sub-windows. But as Lieberman indicates himself, such autonomous interface agents are most appropriate in non-critical decision making situations, especially when the user might be busy with other tasks at the same time.

A somewhat similar project is presented in Payne and Edwards (1997) where the authors discuss the issues involved in the development of an autonomous interface agent, called MAGI, which can filter mail messages. By observing and analysing the user's behaviour in dealing with mail, MAGI can update user preferences and assist her in sorting her mail – a similar application is presented in Malone et al. (1997) and in Maes (1997). Payne and Edwards conclude by noting the need to evaluate interface agents in working environments and to determine if they truly assist users without becoming a nuisance.

Of course, interface agents cannot really assist the user when she needs intelligent on-line help. Once in a while, they can propose useful shortcuts to simplify the user's life in the operation of a frequently used computer application. But that is pretty much all what users can expect from current interface agents.

3.2. *Intelligent assistants and intelligent support systems*

Much work has been done on the development of intelligent assistant systems (Boy 1991). Intelligent assistant systems are characterised by a *co-operative problem solving procedure* and aim at supporting the user's work. Some authors, like Riecken (1997), even propose a clear distinction between agents and assistants: an agent is a simple specialized reasoning process and an assistant is comprised of many co-ordinated agents.

To assist a user's work, her working methods and models should be represented and supported by the system. Many assistant systems leave the control of the problem solving process in the user's hands and take over routine problem solving steps. Hermann (1996) notices that "tools for complex real-world problems are often organized in the wrong way. They try to automate the considered task (or part of it) instead of supporting the problem solving process performed by the user". As several other authors, such as Maes (1994), Hermann (1996) argues that intelligent assistant systems should have learning capabilities in order to better adapt to their users. For example, such a system should learn to avoid an error that occurred once during the problem-solving process, to revise the knowledge base according to selected inconsistencies or weak points, to gain knowledge about problem solving steps from the observation of users' activities, to adapt the knowledge base to the user's problem solving style and to restructure the knowledge base in order to improve comprehensibility.

In Microsoft's Persona Project (Ball et al. 1997), future assistant systems are described as follows (pp. 191–192):

> ... computers will become assistants rather than just tools ... Rather than invoking a sequence of commands which cause a program to carry out small, well-defined, and predictable operations, the user will specify the overall goals of a task and delegate to the computer responsibility for working out the details. ... The machine-like metaphor of a direct manipulation interface is not a good match to the communication needs of a computer assistant and its boss. In order to be successful, an assistant-like interface will need to: Support interactive give and take ... Recognise the costs of interaction and delay ... Manage interruptions effectively ... Acknowledge the social and emotional aspects of interaction.

The Persona Project intends to push the anthropomorphic metaphor to its limits, allowing human-computer interaction to take place through natural (spoken) language processing, dialogue management, and video and audio output in what can be called 'life after the GUI'. The Persona Project certainly constitutes one of the most advanced proposals ever made in the area of conversational interfaces. By all accounts, this project is not only interesting, but quite ambitious and challenging for a variety of practical and theoretical reasons.

The work of Jones and Mitchell (1994) on intelligent support systems is also quite interesting (p. 527): "One way to improve the human-machine system is with intelligent support systems that provide context-sensitive displays, dialogue, and resources for activity management". They discuss the foundations of a supervisory control system, called GT-MOCA, for NASA satellite ground control. Based on four general principles of good human communication (i.e., be relevant, be coherent, appropriately synchronized, and allow for repair), they propose the following design guidelines for intelligent support systems (from Jones and Mitchell 1994, p. 536):

> (i) "Allow the human to retain authority"; (ii) "Support mutual intelligibility (e.g., via the use of inspectable and interactive task and user models)"; (iii) "Be open and honest – the structures and reasoning of intelligent support systems should be inspectable and comprehensible to human users"; (iv) "Support the management of trouble – provide resources (e.g., context-sensitive help at various levels of detail) to help repair breakdowns in communication between human and machine"; (v) "Provide multiple perspectives. . . . Different perspectives can be offered from various levels of abstraction and aggregation".

Jones and Mitchell's proposal takes into account the notion of repair that we consider as highly important. GT-MOCA's repair capability is constructed on two ancillary mechanisms, those of elaboration and explanation, much in the expert system tradition. They also suggest to implement relevance and coherence, referred to as 'mutual intelligibility' above, with the help of user actions monitoring and dynamic modelling of domain goals and user intentions as the basis for co-operative interaction – Levi et al. (1990) also consider that the representation of actions and goals is a critical requirement of intelligent assistant systems. As Smith et al. (1997) put it: ". . . people . . . want help with *their* jobs, *their* tasks, *their* goals".

Mallen (1996, p. 350) argues that an intelligent help system must be able to "answer questions from the user which take account of its current workings and the state of the system" and "provide operational descriptions and explanation þ but in ways which take account of the user's knowledge". And to meet these requirements, Mallen suggests that the help system should: recognise the user's plans, tailor its answers to the user and her current context, deal intelligently with repeated questions, deal with issues of efficiency, and give advice on repair actions. This will require four components: a domain model, a user model, an instructor module, and a communication manager.[8]

We can indeed suppose that the more knowledge sources are used, the more intelligent user interfaces will be. For instance, in the CUBRICON multimedia interface system (Neal and Shapiro 1991), there is a lexicon, a grammar, a discourse model, a user model, a repository of output planning strategies, a repository of general (world) knowledge and a repository of domain-specific knowledge. However, at the current state of the art, the development of realistic systems with such capabilities is still more a goal than an established technology. Nevertheless, such capabilities are definitely worth consideration for the development of future help systems.

4. Recent Trends

Today's human-computer interaction is still very much centred around the keyboard and the screen. But with the elaboration of new concepts like affective computing and with recent progress made in new technologies such as natural language/multimedia interfaces or virtual reality, other interaction means are becoming available or will soon be. We will review some of these new trends here to see how they could contribute to progress towards more helpful interfaces.

4.1. *Multimedia interfaces*

Researchers have investigated the use of several media for supporting the user/system interactions (Maybury 1993): they speak about multimedia or multimodal interfaces. For instance, Maybury and Wahlster (1998) consider voice, gesture, GUIs, natural language and speech, hand gestures, and eye movement.[9] Other researchers have been interested in the use of multimedia in training or education. For instance, Chang and Chen (1995) emphasise multimedia's flexibility in presenting information and providing feedback. Dormann (1996) discusses the interest of on-line animated help instead of plain animations based on simple simulations: she argues that such help mechanisms can reduce the gap between the application and the help system. Van Aalst et al. (1995) say that some topics are very difficult to teach with traditional means and thus suggest that multimedia support productive new approaches. Interestingly, the system they describe, named FLUID, is a multimedia tutorial for user interface design in which the learner plays the role of an assistant user interface designer.[10] Of particular relevance to help systems are projects that use a combination of natural language and realistic objects or maps to express procedural instructions (Feiner and McKeown 1993; André et al. 1993) or to provide explanations (Goodman 1993).

4.2. *Affective computing*

Work at the MIT Media Laboratory illustrates well this relatively new field of computer science research (Picard 1997). Generally speaking, the goal is to endow computers with human-like reactions, perceptions or capacities, in order to allow them to behave in a more natural and sensible manner – a desirable characteristic of believable agents (Bates 1994) and 'charismatic' computers (Fogg 1997). This includes the recognition, modelling and synthesis of emotions, such as impatience and happiness, through direct or indirect use of the computer's input/output devices (keyboard, screen, microphone, speaker), with media like voice/speech, image (e.g., emotional icons (Rivera et al. 1996), faces and facial expression (Essa and Pentland 1997; Cassell et al. 1998; Nagao and Takeuchi 1998)) and text.

In an interesting experiment, Klein et al. (1999) describe a computer system on which a deceiving game is installed. The game makes the user's life miserable by simulating so-called 'random Web delays' which prevent players from performing at the best of their capabilities. This situation quickly leads to frustration which is then measured by a clever experimental set-up as well as (on-line) questionnaires. The researchers found that users were more patient with the frustrating system when they were able to report their problems and feelings to the computer. But the system only allowed users to express themselves about their negative emotional states, it provided them neither with explanations about the frustrating situation nor with solutions that would allow them to avoid or correct the frustrating situation in the future.

In Okada et al. (1999), a task-oriented dialogue system is presented in which emotional aspects are integrated. A rather general model of computational dialogue processing, along the lines of Allen et al. (1996) and Carberry (1990), is augmented with emotion-specific processors that can deal with emotion recognition, emotion arousal, emotion prediction and emotion generation. They use a taxonomy of eight primitive emotions: joy, sadness, acceptance, disgust, surprise, expectancy, anger, and fear. The authors argue that this allows dialogue systems to communicate in a less mechanical fashion which might be more adapted to certain users or certain conversational situations. They also argue that their approach is suitable to endow systems (or agents) with 'personality'. Compared with the work of Klein et al. (1999), this work involves more knowledge intensive processing, using plans and goals specific to emotional aspects. In this sense, it is more typical of artificial intelligence approaches, whereas Klein et al. (1999, p. 4) deliberately stay away from AI, mentioning that: “such a system can be built using existing technology, without requiring strong AI”.

Affective computing makes computers appear more human-like at a superficial level, it does not necessarily make them truly more human-like in any deep sense, and certainly not more human-like at a cognitive level. From our perspective, affective computing provides a pleasant enhancement of a system when the computer (the software) already has all the proper functionalities and knowledge for the task at hand. It does not, for instance, significantly contribute to our needs in terms of designing and developing more intelligent help systems. At best, affective computing techniques would allow an intelligent help system to be more agreeable but not more knowledgeable or more helpful.

4.3. *Virtual reality*

Zheng et al. (1998, p. 20) define virtual reality as follows:

> Virtual Reality (VR) is an advanced, human-computer interface that simulates a realistic environment. ... A virtual reality system should have three characteristics: response to user actions, real-time 3-D graphics and a sense of immersion.

Virtual reality influences the design of computer systems mainly at the level of the user interface. The main argument being that a system using a metaphor that relates ‘naturally’, or ‘intuitively’, to the user and her tasks will be much easier to understand and operate effectively (Shneiderman 1998), Section 6.8). Let us now consider a few examples of such works.

Chu et al. (1997) discuss the problem of determining the requirements for the multi-sensory user interface of a virtual reality based CAD (Computer-Assisted Design) system.[11] Their system uses two essential modes: a navigation mode to navigate through the design space to view generated geometric shapes, and the shape operation mode to create or modify geometric shapes. The model that they propose uses hand and eye motions, voice commands, real-time 3-D images, auditory and tactile feedback. On a related topic, Nadeau (1996) discusses the concept of a 3-D Web, while Wann and Mon-Williams (1996) propose to centre the design of virtual environment systems on the perceptual-motor capabilities of the user, in the context of the task to be undertaken. Similarly, Hodges (1998) presents virtual reality simulations especially designed for training purposes in which trainees could ‘learn by doing’ in environments nearly identical to actual workplaces.

As with affective computing, the realism or intuitiveness that virtual reality brings to user interfaces does not make them more informative or cooperative when users need help. In fact, one could argue that virtual reality is just another way to ease the adaptation of the user to the software, much in the spirit of GUIs, hoping that this will lessen the need for a truly helpful and intelligent help mechanism. The work done by Rickel and Johnson (1999), which is closer to intelligent tutoring systems and agents, is more interesting with regards to help systems. They describe a system called *Steve*, an animated agent, that helps students learn to perform physical, procedural tasks (e.g., how to inspect a high-pressure air compressor aboard a ship). *Steve*'s three main modules, perception, cognition (with domain-specific and domain-independent knowledge) and motor control, allow it to qualify as an AI system – although not as complex as the CUBRICON prototype system (Neal and Shapiro 1991) which was designed to also handle speech – thus making it conceptually distant from the more superficial affective computing or virtual reality systems. Although communication between *Steve* and the user is limited to pre-specified speech utterances, instead of unconstrained natural language, Rickel and Johnson's work suggests a promising direction for further exploration of software help mechanisms. Johnson (2001) presents several examples of so-called *guidebots* (or animated pedagogical agents) which are animated virtual guides which interact with learners in a manner that is inspired by the behavior of human tutors. These interactions include a combination of verbal communication and gestures. Guidebots express thoughts and emotions (enthusiasm, empathy, etc.). These guidebots exemplify the recent convergence of virtual reality approaches, conversation management systems and certain aspects of affective computing that may pave the way to the next generation of help systems.

5. Discussion

Despite the progress made in user-oriented systems presented in the previous sections, we can notice that current help systems and user interfaces continue to suffer from several weaknesses. We drew a list of some of these limitations and discussed the promising areas in current or future research that will eventually be able to deal with them. In what follows, we summarize the main findings of our investigation.

5.1. *Methodological and software engineering aspects*

- We note that although most of the recent methods of software engineering start with the identification of 'use-cases' to orient the design process, there is not much emphasis on the techniques that should be used to develop user-interfaces and to adapt the software functionalities to users' needs. Current software development methods, especially object-oriented ones, which have been widely used in recent years, put little emphasis on the design of user interfaces, users' manipulation needs, and users' help needs. Even notations such as UML (The Unified Modeling Language) seem to neglect this important aspect of software development (Ericksson and Penker 1998). This is an intriguing weakness since the technique of use cases which is used in UML seems to offer the appropriate context to investigate this issue in an in-depth manner and to develop an appropriate user interface. *Let us emphasise the importance of a user-centred software development methodology for the description of the tasks from the user's viewpoint, and the importance of ergonomics (utility and usability) in designing the user interface and the help function.*

- The vast majority of help systems is organized in a way that directly follows a functional view of the software. The trouble with such an approach is that the users' perspective is often drastically different from that of the software designers. *It is important to present a meaningful interface to the user which naturally relates to her view of the software's application domain in terms of the metaphor, functions and objects used.* One way to ensure this is to put the user's needs (manipulation, functional and assistance) at the very centre of the software development process.
- The typical rational used by most software companies seems to be the following: graphic user interfaces allow us to completely control the user's actions in a visually appealing fashion and there is simply no need for complicated help systems. In other words, software should be 'perfect' and the user is expected to adapt herself to the software. And if something goes wrong, if she encounters difficulties or has questions, it is up to her to figure out a way to escape the problematic situation or to discover clues in the on-line help (documentation) system or, worse, in her favourite third-party documentation, in order to go on with her task. *We must insist on the importance of task-specific help and explanations. These have to be adapted to the user's profile and relevant with respect to the user's current situation and goals.* The importance of giving advice on repair actions must also be emphasised. These requirements should be met through the use of a non-intrusive help mechanism, based more on a strategy of user intelligence 'boosting' than user task automation, and supported by a knowledge base expressed in terms of user tasks and goals.

5.2. *Modelling the user*

- Most user interfaces do not exploit a user model. Because of this lack of information, the software is unable to help the user in a truly relevant and timely way and at an appropriate level with respect to her knowledge. To make matters even worse, the vocabulary used in the interface and in the help system is often technical and hardly understandable to a non-specialist user. With such a short-sighted view of the user and her current task, it is simply impossible to assist her intelligently. Hence, the user is expected to ask relevant questions to the system (or to find appropriate items in the help index) and if she fails to do so, or if the help system is unable to answer her request, she is left alone with her frustrations. *It is important to use and maintain a user model in order for the system to behave more co-operatively with the user*. Maybe rather than embarking in the construction of a complex user model based on beliefs, desires, intentions, attitudes, users' mental states, etc., we can adopt a more pragmatic viewpoint which translates into the use and updating of a simple but useful user model. Amongst other things, the user model allows the help mechanism to adapt its assistance to the user's level of expertise. For instance, we expect that tutorial-like assistance will be more appropriate for novice users than for expert users who normally expect assistance more on a demand basis and at a more advanced or technical level. This adaptation is also true for the terminology used in the help facility.
- *We must emphasize how important it is that the user model be developed in a parameterized manner*, allowing it to take into consideration individual user preferences and abilities – this is sometimes referred to as 'personalization' nowadays. We already mentioned the work of Akoumianakis et al. in Section 2.3. There is also the work of Fischer (2000, p. 65) who precisely identifies the challenge facing software developers: "Designers of collaborative human-computer systems face the formidable task of writing software for millions of users (at design time) while making it work as if it were designed for each individual user (only known at use time)".
- The user model can also help to handle some cases of misconception and misunderstanding. *Current user interfaces and help systems are unable to deal with (detect, represent, process) the users' misconceptions*. Such misconceptions can appear for a variety of reasons: lack of knowledge, use of false premises, ambiguous interaction with the software, error during processing, etc. Some of these flaws can be compensated by reactive and proactive system-related explanations.

Mutual understanding can be reached by means of cooperative user-system interaction.

- It is important to design systems that seem to understand what the user wants to do and minimise the user's frustration. *The system must constantly follow the user's objectives and actions*, thus permitting the help system to offer contextually relevant assistance at the right time and, thanks to the user model, at the appropriate level of explanation with a vocabulary adapted to the user. Inversely, the user must be able to follow the system's procedures. The latter must therefore have a proper visual means to inform the user about its state in all circumstances.

5.3. *Agents and other recent trends*

- Most interface agents that have been built up to now, even those that are called "intelligent", are extremely limited – a good number of them being very specific of Web applications – and are usually not designed from the user's perspective of the task. It is at least questionable why such "looking over the user's shoulder" interface agents are so remote from the user's point view or have absolutely no connection with the help system. With regard to interface agents which attempt to learn shortcuts for the user, to perform work on her behalf or to take initiatives with or without her approval, *we believe that it is preferable to rely on a strategy which attempts to amplify the user's intelligence* instead of attempting, with more or less success, to intelligently do her work, or at least some part of it, on her behalf. Such a system would support the user's task in a co-operative manner instead of trying to automate it.
- Recent trends, such as affective computing, conversational interfaces, and some elements of virtual reality, have shown interesting potential for important aspects of human-computer interaction. These generally tend to incorporate more human-like reactions or more realistic interfaces. However, *they still fall short of offering a truly intelligent way to assist the user, as if it had nothing to do with their objectives*. In fact, it is still unclear how these elements could be combined to provide valuable enhancements to future user interfaces. Many of the preliminary results obtained in these fields have led to the production of interface embellishments rather than authentic new tools. It is still not known what types of interactions, or combination thereof, are more appropriate for human-computer interaction in general and, in particular, for help systems. Much more research is needed to push forward these new fields of investigation and to reach more definitive conclusions.

6. Conclusion

Our main concern is that, despite such a large body of knowledge as the one we have attempted to cover here, help facilities found in most commercial software are so ill-conceived that they are usually ‘unhelpful’. We think that human-computer interaction should be seen as a task-oriented co-operative interaction in which the machine is expected to intelligently support the user’s task, especially when she experiences difficulties. The help system must be seen as an integral part of the user interface and its role must be to assist the user in accomplishing her task by whatever suitable means: providing relevant *answers* to her how-to questions about her task or the software’s functionality, and providing useful *advice* on how to perform certain (sub)tasks, but not necessarily to automatically perform them on her behalf. Or, to use Fischer’s wording (2001, p. 65): “say the right thing at the right time in the right way”.

One area of importance for future research is certainly the integration of several of the techniques, tools and approaches covered in this paper. Another area is the study of the “scaling up” problem: many of the proposals presented here are applicable only to small systems or prototype systems: they can hardly be justified on a commercial basis – and this is a serious limitation, quite similar to the one that plagued the development of artificial intelligence systems for many years. Yet another area of interest is that of help-desk systems.[12] Many organizations, especially those selling hardware and software, provide extensive customer support via telephone “help desks”: technical experts must answer the same users’ questions repeatedly. Such support is expensive and many organizations try to reduce their costs by limiting or maintaining the size of their help desk staff and by offering Web-based help desk systems. A search on Internet will show the reader that many companies offer Web-based help desk systems. Certain designers offer guidelines to develop Web-based help systems (See Section 8: Help for the Web initiative), claiming that Web-based help system should provide equal or better the functionality and performance of existing help systems. This new trend raises new questions about help systems: Which services should be available in a software’s help facility and which ones should be supported by a Web-based help desk? Which kind of interaction mode should be offered to the user by either kind of help system? Will Web-based system replace software help facilities? It is intriguing that so little of the developments in this field, which has been relatively active in the last five years, have transferred to actual benefits for help systems. One element of explanation seems to be that many of these systems are based on artificial intelligence techniques (see, e.g., Chan et al. (2000); Fong and Hui 2001; Kang et al. (1997)).

Finally, we think that work on help system would greatly benefit from an in-depth study of today's users' frustrations and expectations when they use their software. It would be very beneficial to get a precise description of the type of feedback and assistance users expect from modern software.

Acknowledgements

This research is supported by the Natural Sciences and Engineering Research council of Canada. The authors wish to acknowledge the contribution of Hengameh Irandoust who carefully revised earlier versions of this paper. Many thanks also to the anonymous reviewers for their helpful comments and suggestions.

Notes

[1] A related idea is that of high-level plans in Bonar and Liffick (1991).
[2] See section VI of Maybury and Wahlster (1998) on 'Model-Based Interfaces' which presents, essentially from an ITS perspective, several tools and techniques that assist the user interface designer.
[3] See Marion (1999c) and Winograd (1996) for a discussion of the importance of user interface in software development.
[4] See our Website addresses on this subject in Section 7.
[5] See our referenced Websites.
[6] See our referenced Website: 'Usable Web: Usability Engineering'.
[7] On ergonomics standards, see Harker (1995) who identifies 'user guidance' and 'guidance on usability' as two important elements.
[8] The reader interested in knowledge-intensive approaches to the design and development of intelligent user interfaces will find representative works in Sullivan and Tyler (1991).
[9] See also the first section of Sullivan and Tyler (1991) on multimodal communication and Burger and Marshall (1998) on the notion of multimedia conversations.
[10] For guidelines on the design of user interfaces to multimedia ITSs, see Najjar (1998).
[11] See also Trika et al. (1997) and Palamidese (1999).
[12] "Help desks are computer aided environments in customer support centers that provide frontline support to external and internal customers": Chan et al. (2000, p. 125).

References

Agah, A. & Tanie K. (2000). Intelligent Graphical User Interface Design Utilizing Multiple Fuzzy Agents. *Interacting with Computers* **12**: 529–542.

AI Magazine (2001). Special issue on Intelligent User Interfaces. *AI Magazine* **22**(4), Winter 2001.

Akoumianakis, D., Savidis, A. & Stephanidis, C. (2000). Encapsulating Intelligent Interactive Behaviour in Unified User Interface Artefacts. *Interacting with Computers* **12**: 383–408.

Allen, J. F., Miller, B. W., Ringger, E. K. & Sikorski, T. (1996). A Robust System for Natural Spoken Dialogue. *Proceeding of the 34th Annual Meeting of the Association for Computational Linguistics Conference (ACL-96)*, 62–70. Santa Cruz, California, USA, 24–27 June.

André, E. & Rist, T. (1993). The Design of Illustrated Documents as a Planning Task. In Maybury, M .T. (ed.) *Intelligent Multimedia Interfaces*, 94–116. AAAI/MIT Press.

André, E. & Rist, T. (2001). Controlling the Behavior of Animated Presentation Agents in the Interface: Scripting versus Instructing. *AI Magazine* **22**(4), Winter 2001: 53–66.

Ardissono, L., Lombardo, A. & Sestero, D. (1993). A Flexible Approach to Cooperative Response Generation in Information-Seeking Dialogues. *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, 274–276. Columbus, Ohio, USA, 22–26 June.

Balbo, S. (1994). Évaluation ergonomique des interfaces utilisateur: un pas vers l'automatisation, Thèse de doctorat en informatique, Laboratoire de Génie Informatique-IMAG, Université Joseph Fourier-Grenoble 1.

Ball, G., Ling, D., Kurlander, D., Miller, J. Pugh, D., Skelly, T., Stankosky, A., Thiel, D., Van Dantzich, M. & Wax, T. (1997). Lifelike Computer Characters: The Persona Project at Microsoft. In Bradshaw, J. M. (eds.) *Software Agents*, chapter 10, 191–222. AAAI/MIT Press.

Bates, J. (1994). The Role of Emotion in Believable Agents. *Communications of the ACM* **37**(7): 122–125.

Bonar, J. & Liffick, B. (1991). Communicating with High-Level Plans. In Sullivan, J. W. & Tyler, S. W. (eds.) *Intelligent User Interfaces*, 129–156. ACM Press.

Boy G. A. (1991). *Intelligent Assistant Systems*. Academic Press.

Bradshaw J. M. (ed.) (1997). *Software Agents*. AAAI Press/MIT Press.

Burger, J. D. & Marshall, R. J. (1998). The Application of Natural Language Models to Intelligent Multimedia. In Maybury, M. T. & Wahlster, W. (eds.) *Readings in Intelligent User Interfaces*, 429–440. Originally published in Maybury (1993), 174–196.

Carberry, S. (1990). *Plan Recognition in Natural Language Dialogue*. The MIT Press.

Carberry, S. (2001). Techniques for Plan Recognition. *User Modelling and User-Adapted Interaction* **11**: 31–48.

Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S. & Stone, M. (1998). Animated Conversation: Rule-Based Generation of Facial Expression, Gesture and Spoken Intonation for Multiple Conversational Agents. In Maybury, M. T. & Wahlster, W. (eds.) *Readings in Intelligent User Interfaces*, 582–589. Morgan Kaufmann. Originally published in *Proceedings of the SIGGRAPH'94 Conference*, 413–420.

Cawsey, A. (1998). Planning Interactive Explanations. In Maybury, M. T. & Wahlster, W. (eds.) *Readings in Intelligent User Interfaces*, 404–419. Morgan Kaufmann. Originally published in *International Journal of Man-Machine Studies* **38**: 169–199, 1993.

Cesta, A. & Romano, G. (1989). Planning Structures for an Intelligent Help System. *Proceedings of the 3rd International Conference on Human-Computer Interaction*, vol. 2, 767–774.

Cetus: A Web site worth 10300 links of object-oriented subjects: http://www.csioo.com/cetusen/software.html

Chan, C. W., Chen, L.-L. & Geng, L. (2000). Knowledge Engineering for an Intelligent Case-based System for Help Desk Operations. *Expert Systems with Applications* **18**: 125–132.

Chandrasekaran, B. (1986). Generic Tasks in Knowledge-based Reasoning. *IEEE Expert* **1**(3): 23–30.

Chang, C.-H. & Chen, Y. (1995). A Study of Multimedia Applications in Education and Training. *Computers in Industrial Engineering* **29**(1–4): 103–107.

Chase, J. D., Parentti, M., Hartson, H. R. & Hix, D. (1993). Task-Oriented User Documentation Using the User Action Notation: A Case Study. *Proceedings of the 5th International Conference on Human-Computer Interaction*, vol. 1, 421–426.

Chin D. N. (1989). KNOME: Modeling What the User Knows in UC. in Kobsa, A. & Wahlster, W (eds.) *User Models in Dialog Systems*. Springer Verlag.

Chin, D. N. (1998). Intelligent Interfaces as Agents. In Maybury, M. T. & Wahlster, W. (eds.) *Readings in Intelligent User Interfaces*. Morgan Kaufmann, pp. 343–357. Originally published in Sullivan, J. W. & Tyler, S. W. (eds.) *Intelligent User Interfaces* (1991), 177–206.

Chin, D. N. (2001). Empirical Evaluation of User Models and User-adapted Systems. *User Modeling and User-Adapted Interaction* **11**: 181–194.

Chiu, C.-T., Chiu, C. & Norcio, A. F. (1993). An Adaptive Intelligent Help System. *Proceedings of the 5th International Conference on Human-Computer Interaction*, vol. 2, 718–723.

Chu, C.-C.P., Dani, T. H. & Gadh, R. (1997). Multi-Sensory User Interface for a Virtual-Reality-Based Computer-Aided Design. *Computer Aided Design* **29**(10): 709–725.

Conger, S. (1994). *The New Software Engineering*. Wadsworth Publishing Company.

David, J.-M., Krivine, J.-P. & Simmons, R. (eds.) (1993). *Second Generation Expert Systems*. Springer Verlag.

Decker, K., Pannu, A., Sycara, K. & Williamson, M. (1997). Designing Behaviors for Information Agents. *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*. Marina del Rey, California, February 5–8.

Dede, C. (1986). A Review and Synthesis of Recent Research in Intelligent Computer-Assisted Instruction. *International Journal of Man-Machine Studies* **24**(4): 329–353.

Dillon, A. & Watson, C. (1996). User Analysis in HCI: the Historical Lesson from Individual Differences Research. *International Journal of Human-Computer Studies* **45**(6): 619–637.

Dormann, C. (1996). Designing On-Line Animated Help for Multimedia Applications. *Lecture Notes in Computer Science*, #1077, 73–84. Springer Verlag.

El-Sheikh, E. & Sticklen, J. (1999). Leveraging a Task-Specific Approach for Intelligent Tutoring System Generation: Comparing the Generic Tasks and KADS Frameworks. *Lecture Notes in Computer Science*, #1611, 809–819. Springer Verlag.

Ericksson, H.-E. & Penker, M. (1998). *UML Toolkit*. Wiley.

Essa, I. & Pentland, A. (1997). Coding, Analysis, Interpretation, and Recognition of Facial Expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(7): 757–763.

Feiner, S. K. & McKeown, K. R. (1993). Automating the Generation of Coordinated Multi-Media Explanations. In Maybury, M. T. (ed.) *Intelligent Multimedia Interfaces*, 117–147. AAAI/MIT Press.

Finin, T. W. (1989). GUMS: A General User Modeling Shell. In Kobsa, A. & Wahlster, W. (eds.) *User Models in Dialog Systems*. Springer Verlag.

Fischer, G. (2001), User Modeling in Human-Computer Interaction. *User Modeling and User-Adapted Interaction* **11**: 65–86.

Fischer, G., Lemke, A. & Schwab. T. (1985). Knowledge-Based Help Systems User Assistance. *Proceedings of the ACM CHI'85 Conference on Human Factors in Computing Systems*, 161–167.

Fogg, B. J. (1997). Charismatic Computers: Creating more Likable and Persuasive Interactive Technologies by Leveraging Principles from Social Psychology, Ph.D. Thesis, Department of Communication, Stanford University.

Fong, A. C. M. & Hui, S. C. (2001). An Intelligent Online Machine Fault Diagnosis System. *Computing & Control Engineering Journal*, October 2001: 217–223.

Friedman, B. & Nissenbaum, H. (1997). Software Agents and User Autonomy. *Proceedings of the First International Conference on Autonomous Agents (Agents'97).* Marina del Rey, California, February 5–8.

Frohlich D., P. Drew & A. Monk (1994), Management of Repair in Human-Computer Interaction. *Human-Computer Interaction* **9**(3): 385–425.

Gardiner, M. M. & Christie, B. (eds.) (1987). *Applying Cognitive Psychology to User-Interface Design*. Wiley.

Gilbert, G. N. (1987): Cognitive and Social Models of the User. *Proceedings of the IFIP INTERACT'87 Human-Computer Interaction Conference*, 165–169.

Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J. & Riedl, J. (1999). Combining Collaborative Filtering with Personal Agents for Better Recommendations. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 439–446. Orlando, Florida, USA, 18–22 July.

Goodman, B. A. (1993). Multi-Media Explanations for Intelligent Training Systems. In Maybury, M. T. (ed.) *Intelligent Multimedia Interfaces*, 148–171. AAAI/MIT Press.

Gould, J. D. & Lewis, C. (1984). Design for Usability: Key Principles and What Designers Think. *Communications of the ACM* **38**(3): 300–311.

Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W. & Harter, D. (2001). Intelligent Tutoring Systems with Conversational Dialogue. *AI Magazine* **22**(4), Winter 2001: 39–52.

Grice, H. P. (1975). Logic and Conversation. In Cole, P. & Morgan, J. L. (eds.) *Syntax and Semantics (Vol. 3, Speech Acts)*, 41–58. Academic Press.

Harker, S. (1995). The Development of Ergonomics Standards for Software. *Applied Ergonomics* **26**(4): 275–279.

Hayes-Roth, B. (1995). An Architecture for Adaptive Intelligent Systems. *Artificial Intelligence* **72**: 329–365.

Hecking, M. (2000). The SINIX Consultant – Towards a Theoretical Treatment of Plan Recognition. In Hegner, S. J. et al. (eds.) Special Issue on Intelligent Help Systems for UNIX, *Artificial Intelligence Review* **14**(3), Kluwer: 153–180.

Hegner, S. (2000). Plan Realization for Complex Command Interaction in the UNIX Help Domain. In Hegner, S. J. et al. (eds.) Special Issue on Intelligent Help Systems for UNIX, *Artificial Intelligence Review* **14**(3), Kluwer: 181–228.

Herrmann, J. (1996). Different Ways to Support Intelligent Assistant Systems by Use of Machine Learning Methods. *International Journal of Human-Computer Interaction* **8**(3): 287–308.

Herrmann, J., Kloth, M. & Feldkamp, F. (1998). The Role of Explanation in an Intelligent Assistant System. *Artificial Intelligence in Engineering* **12**: 107–126.

Hickman, F. R., Killin, J. L., Land, L., Mulhall, T., Porter, D. & Taylor, R. M. (1989). *Analysis for Knowledge-Based Systems (A Practical Guide to the KADS Methodology).* Ellis Horwood.

Hodges, M. (1998). Virtual Reality in Training. *Computer Graphics World* **21**(8): 45–52.

Jameson, A., Paris, C. & Tasso, C. (1997). User Modeling. *Proceedings of the Sixth International Conference UM97.* Vienna, Austria (available on line from http://um.org).

Jennings, N. R. (1999). Agent-Based Computing: Promise and Perils. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1429–1436. Stockholm, Sweden, July 31–August 6.

Johnson, W. L. (2001). Pedagogical Agent Rresearch at CARTE. *AI Magazine* **22**(4), Winter 2001: 85–94.

Jokinen, K. (1996). Reasoning about Coherent and Cooperative System Responses. *Lecture Notes in Computer Science*, #1036, 168–187. Springer Verlag.

Jones, J., Millington, M. & Virvou, M. (2000). An Assumption–based Truth Maintenance System in Active Aid for UNIX Users. In Hegner, S. J. et al. (eds.) Special Issue on Intelligent Help Systems for UNIX, *Artificial Intelligence Review* **14**(3), Kluwer: 229–252.

Jones, P. M. & Mitchell, C. M. (1994). Model-Based Communicative Acts: Human-Computer Collaboration in Supervisory Control. *International Journal of Human-Computer Studies* **41**: 527–551.

Kang, B. H., Yoshida, K., Motoda, H. & Compton, P. (1997). Help Desk System with Intelligent Interface. *Applied Artificial Intelligence* **11**: 611–631.

Klein, J., Moon, Y. & Picard, R. W. (1999). *This Computer Responds to User Frustration (Theory, Design, Results, and Implications)*. Vision and Modeling Group Technical Report #501, MIT Media Laboratory, June 17.

Kobsa, A. & Wahlster, W. (1988). guest editors, Special Issue on User Modelling. *Computational Linguistics* **14**(3).

Kobsa, A. & Wahlster, W. (1989). *User Models in Dialog Systems*. Springer Verlag.

Kobsa, A. (ed.) (2001). Ten Year Anniversary Issue. *User Modelling and User-Adapted Interaction* **11**(1–2).

Komatsu, H., Ogata, N. & Ishikawa, A. (1994). Towards a Dynamic Theory of Belief-Sharing in Cooperative Dialogues. *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, vol. 2, 1164–1169. Kyoto, Japan, August 5–9.

Lashkari, Y., Metral, M. & Maes, P. (1994). Collaborative Interface Agents. *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 444–449. Seattle, Washington, July 31–August 4.

Lecerof, A. (1997). User Interface Evaluation Using Task Models, Master's Thesis, Department of Computer and Information Science, Linköping University.

Lelouche, R. (1998). The Successive Contributions of Computers to Education: A Survey. *European Journal of Engineering Education* **23**(3): 297–308.

Lester, J. C., FitzGerald, P. J. & Stone, B. A. (1997). The Pedagogical Design Studio: Exploiting Artifact-Based Task Models for Constructivist Learning. *Proceedings of the 1997 International Conference on Intelligent User Interfaces*, 155–162.

Lieberman, H. (1997). Autonomous Interface Agents. *Proceedings of the ACM CHI'97 Conference on Human Factors in Computing Systems* (available on line from http://www.acm.org/sigs/sigchi/chi97/proceedings/paper/hl.htm).

Linden, G., Hanks, S. & Lesh, N. (1997). Interactive Assessment of User Preference Models: The Automated Travel Assistant. In Jameson, A. et al. (eds.) *proceedings of the Sixth International Conference UM97*, 67–69. Vienna, Austria.

Liu, Q. & Ng, P. A. (1998). A Query Generalizer for Providing Cooperative Responses in an Office Document System. *Data & Knowledge Engineering* **27**(2): 177–205.

Lovett, M. C. (1998). Cognitive Task Analysis in Service of Intelligent Tutoring System Design: A Case Study in Statistics. *Lecture Notes in Computer Science*, #1452, 234–243. Springer Verlag.

Maes, P. (1994). Agents that Reduce Work and Information Overload. *Communications of the ACM*, Special Issue on Intelligent Agents, **37**(7): 31–40.

Maes, P. (1997). Agents that Reduce Work and Information Overload. In Bradshaw, J. M. (ed.) *Software Agents*, chapter 8, 145–164. AAAI Press/MIT Press.

Mallen, C. (1996). Designing Intelligent Help for Information Processing Systems. *International Journal of Human-Computer Studies* **45**: 349–377.

Malone, T. W., Lai, K.-Y. & Grant, K. R. (1997). Agents for Information Sharing and Coordination: A History and Some Reflections. in Bradshaw, J. M. (ed.) *Software Agents*, chapter 7, 109–143. AAAI Press/MIT Press.

Marion, C. (1999a). User Interface Design (available on line from http://www.chesco.com/∼cmarion/Design/UIDesign.html).

Marion, C. (1999b). Online Information Design (available on line from http://www.chesco.com/∼cmarion/Design/OnInfDes.html).

Marion, C. (1999c). What is Interaction Design (and what does it mean to information designers)? (available on line from http://www.chesco.com/∼cmarion/PCD/WhatIsInteractionDesign.html)

Mathews, M., Pharr, W., Biswas, G. & Neelakandan, H. (2000). An Active Intelligent Assistance Systems. In Hegner, S. J. et al. (eds.) Special Issue on Intelligent Help Systems for UNIX, *Artificial Intelligence Review* **14**(1–2), Kluwer: 121–141.

Maybury, M. T. & Wahlster, W. (eds.) (1998). *Readings in Intelligent User Interfaces*. Morgan Kaufmann.

Maybury, M. T. (1992). Communicative Acts for Explanation Generation. *International Journal of Man-Machine Studies* **37**(2): 135–172.

Maybury, M. T. (ed.) (1993). *Intelligent Multimedia Interfaces*. AAAI/MIT Press.

Mayfield, J. (2000). Evaluating Plan Recognition Systems: Three Properties of a Good Explanation. *Artificial Intelligence Review* **14**: 351–376.

McCoy, K. F. (1983). Correcting Misconceptions: What to Say when the User is Mistaken. *Proceedings of the ACM CHI'83 Conference on Human Factors in Computing Systems*, 197–201.

McCoy, K. F. (1988). Reasoning on a Highlighted User Model to Respond to Misconceptions. In Kosba, A. & Wahlster, W. (eds.) *Computationa Linguistics* **14**(3): 52–63.

McTear, M. F. (1993). User Modelling for Adaptive Computer Systems: a Survey of Recent Developments. *Artificial Intelligence Review* **7**: 157–184.

Miller, J. R., Hill, W. C., McKendree, J., Masson, M. E. J., Blumenthal, B., Terveen, L. & Zaback, J. (1987). The Role of the System Image in Intelligent User Assistance. *Proceedings of the IFIP INTERACT'87 Conference: Human-Computer Interaction*, 885–890.

Mitta, D. A. & Packebush, S. J. (1995). Improving Interface Quality: An Investigation of Human-Computer Interaction Task Learning. *Ergonomics* **38**(7): 1307–1325.

Mo, J. & Crouzet, Y. (1999). A Method for Operator Error Detection Based on Plan Recognition. *Lecture Notes in Computer Science*, vol. 1698, 125–138. Springer Verlag.

Moulin, B. & Chaib-draa, B. (1996). Distributed Artificial Intelligence: an Overview. In Jennings, N. & O'Hare, G. (eds.) *Foundations of Distributed Artificial Intelligence*, 3–55. Wiley.

Moulin, B., Irandoust, H., Bélanger, M. & Desbordes, G. (2002). Explanation and Argumentation Capabilities: Towards the Creation of More Persuasive Agents. *AI Review* (forthcoming).

Myers, B., Hollan, J. & Cruz, I. (eds.) (1996). Strategic Directions in Human Computer Interaction. *ACM Computing Surveys* **28**(4).

Myers, B., Hudson, S. E. & Pausch, R. (2000). Past, Present, and Future of User Interface Software Tools. *ACM Transactions on Computer-Human Interaction* **7**(1): 3–28.

Nadeau, D. R. (1996). User Interface Metaphor in Virtual Reality Using VRML. *Behavior Research Methods, Instruments, & Computers* **28**(2): 170–173.

Nagao, K. & Takeuchi, A. (1998). Speech Dialogue with Facial Displays: Multimodal Human-Computer Conversation. In Maybury, M. T. & Wahlster, W. (eds.) *Readings in Intelligent User Interfaces*, 572–579. Morgan Kaufmann. Originally published in *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, 102–109. Cambridge, Massachusetts, USA, 26–30 June, 1995.

Najjar, L. J. (1998). Principles of Educational Multimedia User Interface Design. *Human Factors – Journal of the Human Factors and Ergonomics Society* **40**(2): 311–323.

Neal, J. G. & Shapiro, S. C. (1991). Intelligent Multi-Media Interface Technology. in Sullivan, J. W. & Tyler, S. W. (eds.) *Intelligent User Interfaces*, 11–43. ACM Press.

Norman, D. A. & Draper, S. W. (eds.) (1986). *User-Centered System Design*. Lawrence Hillsdale, NJ: Erlbaum Associates.

Norman, D. A. (1986). Cognitive Engineering. *User-Centered System Design*, 31–61. Lawrence Hillsdale, NJ: Erlbaum Associates.

Okada, N., Inui, K. & Tokuhisa, M. (1999). 'An Affective Approach to Human-Friendly Dialogue Systems. *Proceedings of the 1999 Conference Pacific Association for Computational Linguistics (PACLING-99)*. University of Waterloo, Ontario (Canada), 25–28 August.

Palamidese, P. (1999). A Virtual Reality Interface for Space Planning Tasks. *Journal of Visual Languages and Computing* **10**: 99–115.

Paris, C. (1988). Tailoring Object Descriptions to a User's Level of Expertise. In Kobsa, A. & Wahlster, W. (eds.) *Computational Linguistics* **14**(3): 64–78.

Payne, T. R. & Edwards, P. (1997). Interface Agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface. *Applied Artificial Intelligence* **11**: 1–32.

Pérez-Quiñones, M. A. & Sibert, J. L. (1996). A Collaborative Model of Feedback in Human-Computer Interaction. *Proceedings of the ACM CHI'96 Conference on Human Factors in Computing Systems* (available on line from http://www.acm.org/sigchi/chi96/proceedings/papers/Perez/map1txt.htm).

Picard, R. W. (1997). *Affective Computing*. MIT Press.

Preece, J. (1994). *Human Computer Interaction*. Addison-Wesley.

Pressman, R. S. (1997). Software Engineering (A Practitioner's Approach), 4th edition.

Quilici, A. (1989). Detecting and responding to Plan-Oriented Misconceptions. In Kobsa, A. & Wahlster, W. (eds.) *User Models in Dialog Systems*. Springer Verlag.

Ramscar, M., Pain, H. & Lee, J. (1997). Do We Know What the User Knows and Does it Matter? The Epistemics of User Modeling. In Jameson, A. et al. (eds.) *Proceedings of the Sixth International Conference UM97*, 429–431. Vienna, Austria.

Rich, C. & Sidner, C. L. (1997). COLLAGEN: When Agents Collaborate with People. *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*. Marina del Rey, California, February 5–8.

Rich, C., Sidner, C. L. & Lesh, N. (1997). COLLAGEN: Applying Collaborative Discourse Theory to Human-computer Interaction. *AI Magazine* **22**(4), Winter 2001: 27–38.

Rich, E. (1998). User Modeling via Stereotypes. In Maybury, M. T. & Wahlster, W. (eds.) *Readings in Intelligent User Interfaces*, 329–341. Morgan Kaufmann. Originally published in *Cognitive Science* **3**: 329–354, 1979.

Rickel, J. & Johnson, W. L. (1999). Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control. *Applied Artificial Intelligence* **13**: 343–382.

Riecken, D. (1997). 'The M System. In Bradshaw, J .M. (ed.) *Software Agents*, chapter 12, 247–267. AAAI Press/MIT Press.

Rivera, K., Cooke, N. J. & Bauhs, J. A. (1996). The Effects of Emotional Icons on Remote Communication. *Proceedings of the ACM CHI'96 Conference on Human Factors in Computing Systems*, vol. 2, pp. 99–100.

Russell, S. & Norvig, P. (1995). *Artificial Intelligence (A Modern Approach)*. Prentice Hall.

Sæbø, K. J. (1988). A Cooperative Yes-No Query System Featuring Discourse Particles. *Proceedings of the 12th International Conference on Computational Linguistics (COLING-88)*, vol. 2, 549–554. Budapest, Hungary, 22–27 August.

Scapin, D. L. (1990). 'Des critères ergonomiques pour l'évaluation et la conception d'interfaces utilisateurs. *Actes du XXVI ème Congrès de la SELF*. Montréal, Québec, Canada, 3–5 October.

Schneider-Hufschmidt, M., Kühme, T. & Malinowski, U. (1993). Adaptive User Interfaces: Principles and Practice. North Holland.

Schreiber, A. Th., Akkermans, J. M., Anjewierden, A. A., de Hoog, R., Shadbolt, N. R., Van de Velde, W. & Wielinga, B. J. (2000). *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press. See also http://www.commonkads.uva.nl/

Schuster, E., Chin, D., Cohen, R., Kobsa, A., Morik, K., Sparck Jones, K. & Wahlster, W. (1988). Discussion Section on the Relationship between User Models and Discourse Models. in Kobsa, A. & Wahlster, W. (eds) *Computational Linguistics* **14**(3): 79–103.

Senach, B. (1990). Évaluation ergonomique des interfaces homme-machine: une revue de littérature. Rapport de recherche INRIA #1180, Programme 8, Communication Homme-Machine.

Sengers, P. (1999). Designing Comprehensible Agents. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1227–1232. Stockholm, Sweden, July 31–August 6.

Shneiderman, B. (1983). Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer* **16**(8): 57–69.

Shneiderman, B. (1997). Direct Manipulation Versus Agents: Paths to Predictable, Controllable, and Comprehensible Interfaces. In Bradshaw, J. M. (ed.) *Software Agents*, chapter 6, 97–106. AAAI Press/MIT Press.

Shneiderman, B. (1998). Designing the User Interface (Strategies for Effective Human-Computer Interaction), 3rd edition. Addison-Wesley.

Shneiderman, B. (1999). *Universal Usability: Pushing Human-Computer Interaction Research to Empower Every Citizen*. HCIL Technical Report No. 99-17 (July 1999), ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/99-17html/99-17.html

Singh, M. P., Rao, A. S. & Georgeff, M. P. (1999). Formal Methods in DAI: Logic-Based Representation and Reasoning. In Weiss, G. (ed.) *Multiagent Systems*. The MIT Press.

Smith, D. C., Cypher, A. & Spohrer, J. (1997). KidSim: Programming Agents without a Programming Language. In Bradshaw, J. M. (ed.) *Software Agents*, chapter 9, 165–190. AAAI Press/MIT Press.

Smith, J. J. (1985). SUSI – A Smart User-System Interface. *Proceedings of the HCI'85 Conference on People and Computers: Designing the Interface*, 211–220.

Sommerville, I. (2000). *Software Engineering*. 5th edition, Addison-Wesley. See also the Web site http://www.comp.lancs.ac.uk/computing/resources/SE6/

Strachan, L., Anderson, J., Sneesby, M. & Evans, M. (1997). Pragmatic User Modelling in a Commercial Software System. In Jameson, A. et al. (eds.) *Proceedings of the Sixth International Conference UM97*, 189–200. Vienna, Austria.

Strachan, L., Anderson, J., Sneesby, M. & Evans, M. (2000). Minimalist User Modelling in a Complex Commercial Software System. *User Modelling and User-Adapted Interaction* **10**(2–3): 109–146.

Sullivan, J. W. & Tyler, S. W. (eds.) (1991). *Intelligent User Interfaces*. ACM Press.

Sutcliffe, A. (2000). On the Effective Use and Reuse of HCI Knowledge. *ACM Transactions on Computer-Human Interaction* **7**(2): 197–221.

Swartout, W. R. & Moore, J. D. (1993). Explanation in Second Generation Expert Systems. In David, J.-M. et al. (eds.) *Second Generation Expert Systems*, 544–585. Springer Verlag.

Takeda, K., Inaba, M. & Sugihara, K. (1996). User Interface and Agent Prototyping for Flexible Working. *IEEE Multimedia Magazine* **3**(2): 40–50.

Tanner, M. C., Keuneke, A. M. & Chandrasekaran, B. (1993). Explanation Using Task Structure and Domain Functional Models. In David, J.-M. et al. (eds.) *Second Generation Expert Systems*, 586–613. Springer Verlag.

Terwilliger, R. B. & Polson, P. G. (1997). Relationships Between Users' and Interfaces' Task Representations. *Proceedings of the ACM CHI'97 Conference on Human Factors in Computing Systems* (available on line from http://www.acm.org/sigs/sigchi/chi97/proceedings/paper/hl.htm).

Trika, S. N., Banerjee, P. & Kashyap, R. L. (1997). Virtual Reality Interfaces for Feature-Based Computer-Aided Design Systems. *Computer-Aided Design* **29**(8): 565–574.

Tsui, K. C. & Azvine, B. (2000). Intelligent Multimodal User Interface. In Azvine et al. (eds.) *Intelligent Systems and Soft Computing*. LNAI 1804, 259–283. Springer.

Van Aalst, J. W., Van der Mast, C. A. P. G. & Carey, T. T. (1995). An Interactive Multimedia Tutorial for User Interface Design. *Computers and Education* **25**(4): 227–233.

van Beek, P., Cohen, R. & Schmidt, K. (1993). From Plan Critiquing to Clarification Dialogue for Cooperative Response Generation. *Computational Intelligence* **9**(2): 132–154.

Virvou, M. (1999). Automatic Reasoning and Help about Human Errors in Using an Operating System. *Interacting with Computers* **11**: 545–573.

Virvou, M., Jones, J. & Millington, M. (2000). Virtues and Problems of an Active Help System for UNIX. In Hegner, S. J. et al. (eds.) Special Issue on Intelligent Help Systems for UNIX, *Artificial Intelligence Review* **14**(1–2), Kluwer: 23–42.

Wahlster, W. (1998). User and Discourse Models for Multimodal Communication. In Maybury, M. T. & Wahlster, W. (eds.) *Readings in Intelligent Yser Interfaces*, 359–370. Morgan Kaufmann. Originally published Sullivan, J. W. & Tyler, S. W. (eds.) (1991). *Intelligent Yser Interfaces*, 45–67. ACM Press.

Wann, J. & Mon-Williams, M. (1996). What does Virtual Reality NEED?: Human Factors Issues in the Design of Three-Dimensional Computer Environments. *International Journal of Human-Computer Studies* **44**: 829–847.

Weiss, G. (1999). *Multiagent Systems (A Modern Approach to Distributed Artificial Intelligence)*. The MIT Press.

Weissert, C. (1996). *Increased User Satisfaction through Improved Error Messages*. Master's Thesis, Department of Computer Science, California State Polytechnic University.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. Morgan Kaufmann Publishers.

Wick, M. R. & Thompson, W. B. (1992). Reconstructive Expert System Explanation. *Artificial Intelligence* **50**: 33–70.

Winograd, T. & Flores, F. (1986). *Understanding Computers and Cognition (A New Foundation for Design)*. Reading.

Winograd, T. (ed.) (1996). *Bringing Design to Software*. Addison-Wesley.

Ye, N. (1997). Neural Networks Approach to User Modelling and Intelligent Interface: A Review and Reappraisal. *International Journal of Human-Computer Interaction* **9**(1): 3–23.

Young, R.L. (1991). A Dialogue User Interface Architecture. In Sullivan, J .W. & Tyler, S. W. (eds.) *Intelligent User Interfaces*, 157–176. ACM Press.

Zheng, J. M., Chan, K. W. & Gibson, I. (1998). Virtual Reality (A Real World Review on a Somewhat Touchy Subject). *IEEE Potentials* **17**(2): 20–23.

Zhou, Y., Freedman, R., Glass, M., Michael, J. A., Rovick, A. A. & Evens, M. W. (1999). Delivering Hints in a Dialogue-Based Intelligent Tutoring System. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 128–134. Orlando, Florida, USA, 18–22 July.

Zukerman, I. & McConachy, R. (1995). Generating Discourse across Several User Models: Maximizing Belief while Avoiding Boredom and Overload *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, vol. 1, 1251–1257. Montréal, Québec, Canada, August 20–25.

Web Sites

AIRG AI Webliography
http://www.cs.wpi.edu/Research/airg/AI-hotlist.html

3D Site: Virtual Reality
http://www.3dsite.com/n/sites/3dsite/cgi/virtual-reality-index.html

ACM SIGCHI (Special Interest Group on Computer-Human Interaction)
http://www.acm.org/sigchi/

Agent Architectures
http://www.cs.buffalo.edu/∼goetz/agents.html

Association of researchers working on User Modelling
http://www.um.org/

Ben’s VR (Virtual Reality) Links
http://www.cc.gatech.edu/gvu/people/Phd/Benjamin.Watson/links/vr.html

Bibliography of the International Journal on User Modelling and User-Adapted Interaction
http://liinwww.ira.uka.de/bibliography/Ai/UMAI.html

Craig Marion’s User Interface Design
http://www.chesco.com/∼cmarion/Design/UIDesign.html

Help for the Web Initiative
http://www.help4web.org/index.html

Human-Computer Interaction Resources on the Net
http://www.ida.liu.se/labs/aslab/groups/um/hci/

Intelligent Software Agent Bibliography
http://ils.unc.edu/gants/agentbib.html

Intelligent Software Agents
http://www.sics.se/isl/abc/survey-main.html

Intelligent User Interfaces
http://excalibur.brc.uconn.edu/∼aui/iui3.html

Linguistics, Natural Language, and Computational Linguistics Meta-index
http://www.sultry.arts.usyd.edu.au/links/linguistics.html

M.E.P. Plutowski’s Emotional Computing Book
http://www.emotivate.com/Book/index.htm

MIT Media Lab: Affective Computing
http://affect.www.media.mit.edu/projects/affect/
Multiagent Systems Research Group
http://www.cs.wustl.edu/∼mas/links.html
Psychology & Design Web links: Cognitive Ergonomics
http://www.ntu.ac.uk/soc/bscpsych/psydes/cogerg.htm
Research at the MIT Media Laboratory
http://www.media.mit.edu/Research/
The Association for Computational Linguistics
http://www.aclweb.org/
The Cognition And Affect Project (School of Computer Science, University of Birmingham)
http://www.cs.bham.ac.uk/∼axs/cog_affect/COGAFF-PROJECT.html
Usability Engineering Web Resources
http://inf2.pira.co.uk/jzus1.htm
Usability Guide for Software Engineers (GUSE)
http://www.otal.umd.edu/guse/
Usability Links for Web design
http://www.usableweb.com/
Usable Web: Usability Engineering
http://usableweb.com/items/usabilityeng.html
User Interface Design Bibliography
http://world.std.com/∼uieweb/biblio.htm
User Interface Design: collection of URLs
http://www.chesco.com/∼cmarion/Design/UIDesign.html
Virtual Reality Links
http://www.loria.fr/equipes/isa/pointers.html
Workshop on Adaptive Systems and User Modelling on the World Wide Web
http://www.contrib.andrew.cmu.edu/∼plb/WWWUM99_workshop/