

Intelligent interaction design: the role of human–computer interaction research in the design of intelligent systems

Ann Blandford

School of Computing Science, Bounds Green
Road, London N11 2NQ, UK
E-mail: A.Blandford@mdx.ac.uk

Abstract: *As more intelligent systems are introduced into the marketplace, it is becoming increasingly urgent to consider usability for such systems. Historically, the two fields of artificial intelligence (AI) and human–computer interaction (HCI) have had little in common. In this paper, we consider how established HCI techniques can usefully be applied to the design and evaluation of intelligent systems, and where there is an urgent need for new approaches. Some techniques — notably those for requirements acquisition and empirical evaluation — can usefully be adopted, and indeed are, within many projects. However, many of the tools and techniques developed within HCI to support design and theory-based evaluation cannot be applied in their present forms to intelligent systems because they are based on inappropriate assumptions; there is consequently a need for new approaches. Conversely, there are approaches that have been developed within AI — e.g. in research on dialogue and on ontologies — that could usefully be adapted and encapsulated to respond to this need. These should form the core of a future research agenda for intelligent interaction design.*

Keywords: artificial intelligence, human–computer interaction, usability, dialogue, ontologies, model-based evaluation, intelligent user interfaces

1. Introduction

As more and more of the techniques developed within the field of artificial intelligence (AI) mature to the point where

they are incorporated within production-standard systems — i.e. commercially available systems that are intended to be used by people who have no background in AI — there is an increasing need to consider usability issues for such systems. Systems that are difficult or inefficient to use are prone to errors, alienating and costly to run and maintain. Many systems appear to be developed for use with no form of usability evaluation. For example, Lee and Evens (1998) make usability claims about their approach to natural language input to an expert system but make no reference to any form of usability evaluation. Similarly, Xu and Li (2000) present an approach to integrating a number of paradigms (rule-based, case-based and model-based reasoning) to effectively solve a problem, and discuss the acceptability of the resulting solutions, but do not comment at all about how users are expected to interact with the system, or whether or not they might choose it over any alternative. The most notable exception is in the world of intelligent tutoring systems (ITSs), where the focus is generally on educational effectiveness, which cannot be achieved without interactional effectiveness. As Self (1999) puts it: ITSs care about their users — or at least, their designers do.

Each generation of computing goes through the same phases — of dealing with technical challenges first while limiting use to specialists, then releasing systems for more general use, and only later considering the needs of these new groups of users. Indeed, it is only with the advent of the World Wide Web and the growth of electronic commerce (Lohse, 2000) that the importance of usability has been widely recognized. Historically, users have had to invest money in software before they could effectively test its usability, and then had to invest time in learning how to use it or abandon it as a waste of money. While users, paradoxically, often gain substantial satisfaction from the sense of competence afforded by their ability to use difficult systems (Landauer, 1995, p. 191), this represents an unjustifiable waste of resources. In contrast, when accessing Web pages over the Internet, user investment is very small and there is little incentive to master a difficult site, so if a Web site is difficult to use the user is liable to leave quickly, and the site owners will have failed to achieve their objectives (making a sale or imparting information). This shift in the point of end-user commitment to working with a particular system has forced developers to pay more attention to usability and usefulness. As more intelligent systems are taken up and used commercially, it will be essential that they, too, are usable.

Although in practice the boundaries are blurring between ‘traditional’ and ‘intelligent’ systems, for the purposes of

this paper they will be treated as two distinct types of system. As this paper concerns usability, it might be tempting to start from a user perspective on what would make a system class as 'intelligent'. However, any definition that starts from this viewpoint is likely to be flawed: many traditional systems can appear unpredictable, even autonomous, to a user who has little understanding of their operation, so a system might appear to be 'intelligent' when in fact it is simply inscrutable. Therefore, we take a textbook definition of AI, as the 'branch of computer science that is concerned with the automation of intelligent behaviour' (Luger & Stubblefield, 1998). Thus AI covers systems that learn, plan and reason. We will take AI to encompass technologies that some might now consider nearly 'traditional' such as rule-based expert systems as well as less well understood technologies such as those based on advanced neural nets or genetic algorithms; however, in this paper, the focus of concern is not the underlying technology, but the system behaviour. Generally, intelligent systems have been ignored by the human-computer interaction (HCI) community, except where they can be assumed to raise the same usability issues as traditional systems. The only real exception to this is those systems that would class as 'intelligent user interfaces' (e.g. Pilkington, 1992; Jameson *et al.*, 1999), including interface agents (Laurel, 1990; Lester *et al.*, 1997; Lieberman, 1997; Rich & Sidner, 1997; Johnson *et al.*, 2000); in these cases, the focus has generally been on what is possible, rather than on what is desirable — a point that is expanded on in the next section.

2. An overview of traditional HCI

The ultimate aim of work on HCI is the design and implementation of effective interactive systems, involving computers and users. This necessarily involves work that has traditionally developed separately in various disciplines, including computer science, psychology, design theory, social sciences, work domain analysis and creative design. HCI is not so much a discipline, with a tightly defined set of research methods and practical techniques, as a meeting point for the exchange of ideas, theories and approaches.

We start by proposing a framework for understanding the nature of work in HCI, so that AI systems and practices can be related to this framework.

2.1. A framework for HCI

HCI can be characterized in terms of two dimensions, as shown in Table 1. The first dimension concerns the stage in the design process: generation or evaluation. Here, we use the term 'generation' to mean the development of solutions to a design problem, and 'evaluation' to mean the assessment of possible solutions against the original problem to see how well they match. In a traditional design

lifecycle (see, for example, Wilson *et al.*, 1989; Avison & Fitzgerald, 1995), most 'generation' activity takes place in the early stages of the lifecycle, and 'evaluation' in later stages. In more recent approaches to design that accommodate user perspectives from early on, and are typically iterative (e.g. Preece *et al.*, 1994; Mumford, 1995), the activities of generation and evaluation are more closely intertwined. Nevertheless, different skills and techniques are needed for each phase. Generation is a creative process in which novel solutions are proposed, described, developed and implemented. Evaluation is an analytical process in which critical questions are asked and proposed solutions are tested against requirements.

The second dimension of contrast concerns the type of approach taken: theory-based or empirical. Theory-based approaches to design typically result in small-scale design evolution: theories normally exist for well-studied situations, and cannot be extended far outside them, so they are constrained to support only relatively minor changes from earlier designs. Theory-based approaches to evaluation draw on what is understood, and appropriately expressed, within the contributing disciplines; one example is work in cognitive science that can be applied to reason about likely user behaviour with a computer system even before that system has been implemented (e.g. Gray *et al.*, 1993; Wharton *et al.*, 1994; Blandford & Young, 1996). In contrast, empirical approaches to design generation result in more novel, revolutionary design solutions — design solutions that might change the way we think and work or, conversely, might be considered failures, fading without trace. Dix *et al.* (1998) summarize many of the important design paradigms that might be classed as 'design revolutions', including the graphical user interface, the World Wide Web and agent-based interfaces. Empirical approaches to evaluation draw on quantitative testing techniques from psychology (traditional experimental design, which yields results that can be evaluated using statistical techniques) and qualitative techniques, such as the use of interviews and questionnaires, and the analysis of verbal protocols (Ericsson & Simon, 1984; Strauss & Corbin, 1998) from the social sciences.

As shown in Table 1, most existing work on intelligent systems is empirical/generation: AI develops theories of intelligence and applies those theories in the design, implementation and testing of systems, but the theories being developed and tested are not theories of users or interactions or of how the system will be used; they are theories of representation or novel algorithms that, when applied, yield novel systems. This view is consistent with that of Partridge and Wilks (1987), who characterize the typical AI development lifecycle as consisting of four stages: run — understand — debug — edit. According to this view, it is only through the process of developing and running a computer system that the programmer acquires an understanding of its behaviour, so the primary role of

Table 1: *Characterizing HCI*

	<i>Generation</i>	<i>Evaluation</i>
Theory-based	Evolutionary design, based on existing paradigms and established software development methods; the focus is on what is needed rather than pushing the boundaries of what is possible	Usability assessment of specification or implementation, based on established theories
Empirical	Novel/revolutionary design, often based on novel theories (e.g. in AI technologies); the focus is on what is possible more than what is needed	Usability assessment of prototype by testing with representative users

the implementation is as an environment for developing and testing theories, rather than to be used in earnest by others. Indeed, such systems are generally difficult or impossible for others to use. Developers of production-standard systems need less exploratory design methods.

Theories for design generation are gradually emerging for the more established AI-based technologies such as knowledge-based and expert systems. For example, various authors have proposed design lifecycles for knowledge-based systems. Wilson *et al.* (1989) compare knowledge engineering with traditional software engineering, and argue that it is more iterative because the requirements are less well understood, and therefore less explicitly defined at the beginning of the process. Partridge and Hussain (1995) attempt to create a more linear, systematic development lifecycle for knowledge-based systems, but accommodate iterations at many levels. In particular, they suggest that each component of a substantial system is likely to go through an extensive development, test and redesign cycle.

Reports of user testing of AI-based systems are rare, but such work would class as empirical/evaluation. One example is the work of Bertin *et al.* (1998), who present an account of their approach to including evaluation throughout the process of design, from prototyping through design to full development. Their description emphasizes the need for early prototyping as a means for conducting formative evaluation that can be fed back into the design process. Another, contrasting, example is the work of Cawsey *et al.* (2000), who use a statistical experimental approach to assess the effect of personalization of information. They measured users' responses to information under personalized and non-personalized conditions to yield a summative evaluation of the effect of personalization.

To date, little work on intelligent systems has transparently taken a theory-based approach to design evaluation (where here it should be emphasized that the theory in question is HCI theory, rather than AI theory). The focus of this paper is on theories that already exist, and could be used with or without adaptation. We also identify gaps where additional theories, methods and tools are needed.

2. 2. *Systems thinking*

Effective interaction design involves considering a system at many different levels of detail. A new computer system has to work at all the different levels to be truly usable and useful. For each level, as discussed in the following section, we can identify a range of requirements for usability. First, we briefly discuss the levels.

Figure 1 shows a schematic representation of a system at the level of users, computer systems and other artefacts in the work setting. Most work in computer science and AI focuses on the design of the individual computer system or computer networks. Most work in HCI focuses on the interaction between a single user and their computer system (signified by the darker shaded components in Figure 1), but can extend to consider the wider system within which the user and computer system reside. For example, a particular computer system has to fit in with the user's existing working patterns, other tools and the context of use (Beyer & Holtzblatt, 1998). In any substantial organization, knowledge and responsibilities are typically distributed among the agents (people, computer and other objects in the work system). Work on distributed cognition (Hutchins & Klausen, 1991) aims to give an account of how roles, responsibilities and resources are distributed around a multi-agent system, and how each agent maintains awareness of important aspects of the state of the overall system. Applying this specifically in the HCI context, Wright *et al.* (2000) model the ways various resources in the environment mediate communication and serve as external memory aids, to support reasoning about the roles of both computer-based and other artefacts in supporting work.

As is argued by Hollnagel and Woods (1983), an interactive system cannot be completely understood just in terms of its constituent components: the whole is more than the sum of the parts. Users adapt their behaviour with a computer system according to both their goals and their understanding of how the computer system works and what state it is in. Conversely, the machine embodies a representation, explicit or implicit, of the user. In the case of

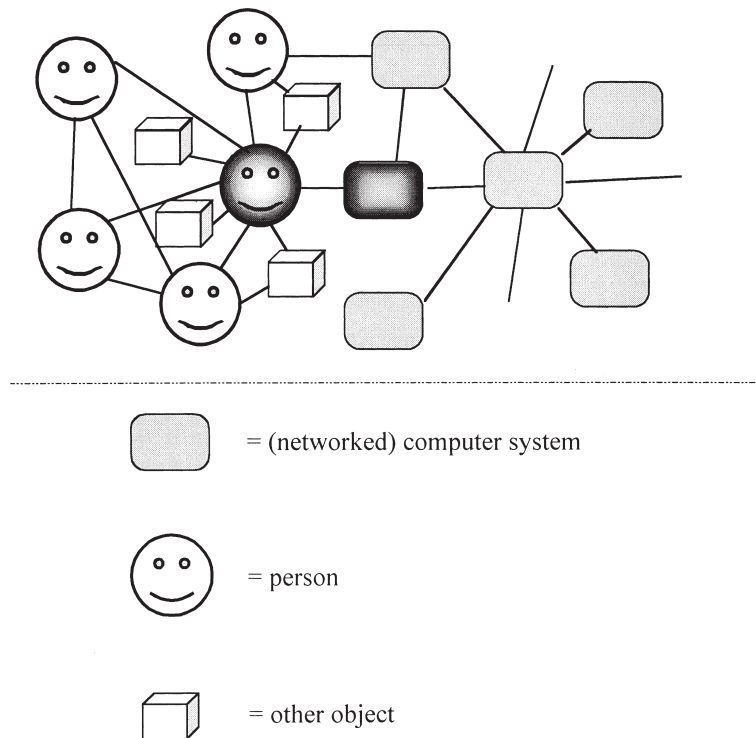


Figure 1: *A system as interacting agents and objects.*

adaptive systems this may be in the form of an explicit user model; in other cases it might be an implicit set of assumptions made by the designer about how the users are intended to behave, and what they are expected to know about the behaviour and state of the machine.

2. 3. Requirements on interactive systems

Hackos and Redish (1998, p. 6) specify a fairly simple set of requirements on the design of a system: that it should 'reflect the workflows that are familiar or comfortable', that it should be 'compatible with the users' working environment' and that it should be easy to learn. They unpack this last requirement further — that it should support users with different learning styles, that it should employ a familiar metaphor or idiom, that the presentation should be consistent, and that the language and illustrations used should be familiar or easy to learn.

Other authors, explicitly or implicitly, focus on requirements that are largely consistent with those of Hackos and Redish, emphasizing some over others. For example, Gould (1988) focuses on the need for reliable computer systems, while Johnson (1992) emphasizes the importance of understanding users' tasks and designing to support them.

This idea that the computer system is supporting the user in performing some tasks — or achieving some goals — is central, but does not fit particularly naturally with the characterization of a system presented above in terms of

levels. Any user has a conceptualization — a more or less clearly articulated understanding — of the domain tasks in terms of the procedures that are followed and the entities, both physical and intangible, that 'define' the domain. One of the requirements on a computer system that supports this task is often that it should do so 'transparently': that the representation the user manipulates via the computer interface should have a natural mapping to its real-world counterpart.

Table 2 outlines a range of typical requirements on the design of any interactive system.

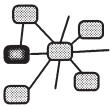

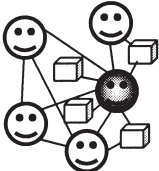
3. Understanding HCI requirements with respect to intelligent systems

We consider each of these views of 'system', focusing initially on traditional HCI theories, methods and tools and then, under each heading, considering to what extent the existing techniques can be applied to AI systems and identifying some of the important gaps.

3. 1. Computer systems

Extensive descriptions of the most widely available interaction technologies are provided in many HCI texts (e.g. Shneiderman, 1998). The same basic range of technologies is available for AI systems. For example, many systems exploit standard graphical user interfaces or simple text

Table 2: *Requirements that apply to the system at different levels*

	Level	Requirements	Section
	Computer systems	Computer system is functionally correct and reliable Visual structures support task easily	3.1
	Interactive system and work system	User errors are minimized User can learn easily User's conceptual model fits well with that implemented within the device User's device task structure maps naturally onto domain task structure	3.2
	Context of use and people	Device fits into the context of use Computer systems effectively support group working	3.3

entry (e.g. Partridge & Hussain, 1995, p. 155). Indeed, one common requirement for usable AI systems (e.g. Bertin *et al.*, 1998) is that they should run on widely available hardware platforms.

3.1.1. Computer systems must be functionally correct and reliable As Gould (1988) observes, a computer system that is not reliable is not usable. Techniques for system development range from rapid prototyping to formal methods (e.g. Hall, 1990) to ensure software reliability. Two aspects of correctness are often discussed separately: verification ('have we built the system right?') and validation ('have we built the right system?'). These are central to dependable systems design, but cannot work without effective requirements acquisition and understanding the context of use, as discussed below.

Verification addresses questions such as: has all the required functionality been provided within the system, and is the system reliable and trustworthy? Within non-user-oriented software engineering, verification generally focuses on functional requirements, rather than on non-functional requirements that are concerned with how the system will actually be used, whether it will provoke user errors, whether it will be pleasurable and efficient to use etc. Techniques that take account of the users within this process are gradually being developed (e.g. Hollnagel, 1998).

Validation addresses questions about the appropriateness

of the system, e.g. has the most appropriate functionality been identified, and has it been implemented in a way that fits well with the context of use? These questions are necessarily more oriented towards users and use.

Design methods for knowledge-based systems typically focus more on knowledge acquisition, i.e. getting domain knowledge from experts rather than understanding requirements of future users of the system. Design methods for other intelligent systems typically adopt the 'run — understand — debug — edit' approach discussed above; again, user requirements are rarely stated explicitly, so that it is difficult to evaluate the correctness of a resulting system against any user requirements. Fox *et al.* (1998) observe that 'There is a long-established orthodoxy within the expert system community that expert knowledge is intrinsically informal (heuristic) and expert systems are consequently not amenable to the use of formal design techniques.' While verification techniques for intelligent systems are likely to be very different from those for traditional ones, validation techniques should, at least in principle, be similar. Assessing to what extent a system as designed serves the intended purpose should be independent of the style of implementation.

3.1.2. Visual structures support the task easily It has long been recognized that user interpretation of information

on a graphical display may depend on the precise terminology used, the relative locations of items on the screen and other features such as the colours or icons used.

Techniques for identifying appropriate terminology include 'Wizard of Oz' (Dahlbäck *et al.*, 1993), in which the user interacts with an experimenter via a terminal to establish how they most commonly phrase queries and requests and what terms they use for core concepts within the domain of application. The core issues concerning identification of suitable terminology are exactly the same for intelligent and for traditional systems. Pilkington (1992) describes the use of this method in detail for not only identifying terminology but also identifying requirements on dialogue structures and recognizing user intentions for the design of help dialogues. The technique can also be used for determining the most natural order for queries in simple (non-intelligent!) spoken-language dialogues such as telephone-based menu selection systems (though experience shows that there currently exist many systems in use that fail the 'naturalness' criterion). The latter is an example of using Wizard of Oz to support task analysis, a topic that is discussed further below.

As well as using appropriate terminology, the effective use of graphics and animation is important. Much of the work on the psychology of vision and interpretation of visual structures has focused on comprehension of ambiguous visual stimuli and other low-level phenomena that have little relevance to work on HCI. However, work on the ways grouping of objects can affect their interpretation and how the substructures of objects can affect their discriminability (May *et al.*, 1993) are equally relevant to the design of traditional and intelligent systems. For example, if there were a selectable item labelled 'date', the interpretation would be likely to be strongly influenced by whether it was located near other items labelled 'banana' and 'nut' or items labelled 'time' and 'month'. Also, how people interpret information across the 'cut' when scenes change abruptly (May & Barnard, 1995) is relevant to the design of any dynamic display: an obvious example would be the case of animated agents that appear on the screen to help users with their work, whose movements should be related to other things going on in the display.

3.1.3. Summary: computer system requirements While the underlying theories on which systems are built differ between traditional and intelligent systems, at the low level, considering input and output technologies, graphical structures, terminology used etc., the issues and theories are much the same, and designers of intelligent systems have a ready resource in existing work within HCI.

3.2. Interactive and work system

As noted above, one of the most commonly stated requirements on an interactive system is that it should be easily

learned. As well as being learnable, there are usually other requirements, e.g. that user errors are minimized. Since the user is (generally) working on some domain task, and understanding what they are doing in terms of that task, the device also has to present an appropriate representation of the task that fits well with the user's conceptualization — both structurally and procedurally. We consider each of these requirements in turn.

3.2.1. User can learn easily All systems should be learnable. In some cases — notably with 'walk up and use' devices such as automatic teller machines — the learning required should be minimal: the device should be self-explanatory, assuming minimal prior knowledge on the part of the user. In other cases — e.g. games — the learning may be made intentionally difficult to increase the challenge to the user. However, most systems fall somewhere between these extremes. The requirement that a system is learnable can be considered from many perspectives, and a variety of techniques have been developed to assess different aspects of learnability.

Norman (1986) proposes a seven-stage model of interaction that provides one useful perspective on learnability. The stages are as follows:

1. Goal formation: the user acquires a domain goal (e.g. to make a future event memorable).
2. Intention formation: the user forms intentions about how to achieve the goal (e.g. adding an 'event' to an electronic time management system).
3. Action specification: the user determines the next action to perform (e.g. to scroll to the place in the time management system that corresponds to the time of the event), or forms a plan consisting of several actions.
4. Action execution: the user executes the next action in the plan (presses the scroll key).
5. Perception: the user sees the change in the state of the device (e.g. a new spatial display of events has appeared on the screen).
6. Interpretation: the user interprets the new system state (e.g. we have or have not yet scrolled to the desired point).
7. Evaluation: the user evaluates the new state against their goals (e.g. yes, the correct place has been reached, and the next step is to enter the event information).

Difficulties in stages 2 and 3 contribute to what Norman calls the 'gulf of execution': users have to use their knowledge of the device to identify actions that are likely to make progress towards their domain goal, and that knowledge has to be learnt. Difficulties in stages 5 to 7 contribute to the 'gulf of evaluation': users have to be able to interpret perceived changes in state in terms of their goals and domain

knowledge. This applies whatever underlying technology the system is based on.

Norman proposes that an important role for the interface is to project the designer's conceptual model of the underlying system to the user, so that the user can acquire an accurate, though not necessarily complete, conceptual model of that system, as illustrated in Figure 2. As Hollnagel and Woods (1983) would see it, designers have to use their understanding of the user to help develop an appropriate representation for the user. However, one of the difficulties with this, as Landauer (1995) points out, is that unless designers establish close contact with users and make a point of studying how they work, they tend to rely on their intuitions of how users will behave and understand things, and those intuitions can often be highly inaccurate.

Note that this view puts the designer at the centre: the designer's job is to enable the user to assimilate the designer's view by getting the device to project an appropriate image of itself. This is necessary for device-only constructs, where the challenge is to make these as easy to comprehend and learn as possible, but rather ignores the centrality of the users' domain knowledge in their interpretation of information from the display. This idea that users acquire an internal representation of a system that can be interrogated and manipulated, and is used as a basis for their interaction with that system, is embodied in work on mental models (Johnson-Laird, 1981). Mental models are constructed from experience and are generally incomplete and inaccurate, but serve a purpose. In particular, they can be 'run' as mental simulations of the effects of actions as a means of deciding what to do, and of interpreting the effects of actions. Much of the work on mental models has been conducted on devices that define or create their own domains — e.g. pocket calculators (Young, 1981) support a task that was previously done using pencil and paper, but the usability of a calculator depends on the user's ability to acquire and use an understanding of the 'memory' of the

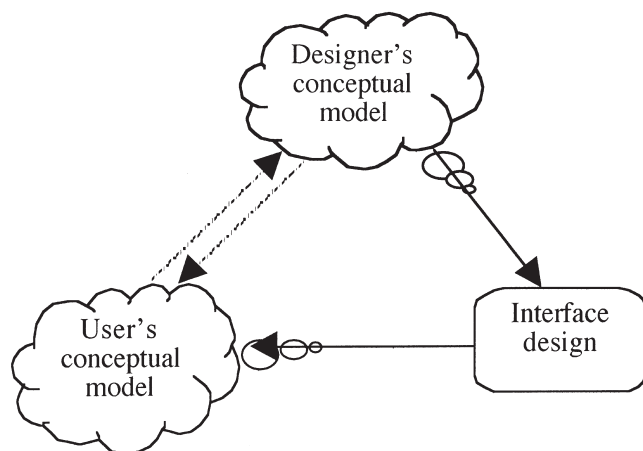


Figure 2: *The designer communicates with the end user through the interface.*

device, a feature that does not mimic paper and therefore has to be learnt for the device.

This kind of thinking is highly pertinent for intelligent systems that serve as a working environment within which the user can investigate some phenomenon. Many spreadsheets, decision support systems and programming environments would fall in this class, and most investigative AI work is conducted within this paradigm. For example, a user changing parameters within a neural net to obtain the best possible model of some real-world phenomenon, or a user manipulating data using a self-organizing map (Kohonen, 1984), needs to know enough about how it works to be able to manipulate it effectively. Du Boulay *et al.* (1981) refer to such systems as 'glass boxes' and, in the context of teaching novice programmers, discuss how the user can be supported in acquiring an appropriate understanding of the device. The most important consideration is that the user's mental model of the system has to be adequate to support effective use. One central issue for designers of such a system is how to communicate their conceptual model of the system to the user.

Within traditional HCI, there have been extensive discussions on what abstract features of a design make a system learnable. For example, Dix *et al.* (1998) propose a set of principles that a system should satisfy if it is to be learnable, namely that it should be

- predictable, supporting the user in determining the effects of future actions based on past experience of working with this device;
- synthesizable, supporting the user in assessing the effects of past actions on the current state;
- familiar, allowing users to apply their knowledge and experience of other real-world or computer-based domains when interacting with the current system;
- generalizable, allowing users to extend their knowledge from one situation to others, within and across applications;
- consistent, having similar input and output behaviour in similar situations.

As well as focusing on the need for the device to communicate its behaviour to the user, so that the user can acquire a mental model of it, Dix *et al.* introduce principles that allow the user to relate the behaviour of this particular device to other experiences.

Learnability as presented by Dix *et al.* might, at first sight, appear to be irrelevant to the design of many intelligent systems. For example, it is not possible for the user of an adaptive system to develop a simple mental model of its operation. However, consider interaction with an information system such as the World Wide Web. The interaction is not predictable in a full sense, but that is the way it should be: if the consequences of following a link were predictable there would often be little point in using the Web. Problems arise for a Web user when it is not

predictable in a broader sense: that the user does not know how the browser works, or is surprised by the effects of (for instance) pressing the navigation keys (Cockburn & Jones, 1996). Similar arguments apply to the other aspects of learnability. In such a case, the user does not need a detailed understanding of the system, but does need to understand some of the principles underlying its design. This is a more appropriate place to start thinking about learnability of an intelligent system: not that it should be learnable in detail, but that it should be comprehensible and make sense to the user, and that the user should be able to work out what the most sensible next actions are, i.e. that the 'gulfs' are as small as possible between user and system.

Another high-level approach to assessing usability has been developed by Nielsen (1994), who proposes a series of ten questions that should be asked about any interface design. Many of the questions encapsulate the same ideas as Dix *et al.*'s principles. However, some extend to cover issues such as how easily a user will be able to recognize and recover from mistakes — or how mistakes might be avoided in the first place. Such approaches tend to focus on superficial aspects of the interface design, and rely substantially on the expertise of the evaluators for the identification of any deeper difficulties.

Existing approaches that support deeper analysis are often based on cognitive models of users. They include techniques such as GOMS (Card *et al.*, 1983; John & Kieras, 1996), cognitive walkthrough (Wharton *et al.*, 1994) and PUM (Blandford & Young, 1996). These techniques are all based on the 'model human processor' (Card *et al.*, 1983) view of cognition — that the user receives stimuli from the outside world, processes that information, and forms goals and plans how to act on the basis of that information. GOMS involves defining user behaviour in terms of procedures they are expected to follow when using a device; it assumes the user is an expert, with perfect knowledge of the device. Conversely, the cognitive walkthrough technique is based on the idea that the user is learning about the interface in an exploratory way, and has goals, and is applying simple means-ends reasoning to decide what to do next. PUM focuses on identifying the knowledge a user needs to work with a device at a fine grain of detail, and establishing where the user is expected to acquire that knowledge from; one of the original premises of the PUM work (Young *et al.*, 1989) was that if the knowledge could not be laid out in such a way that a simple planner (e.g. Fikes & Nilsson, 1971) could use the defined knowledge to determine what to do next then real users would be likely to have difficulty too. Although these techniques are derived from early work in AI, and might be expected to take more account of intelligent systems, in practice they all focus entirely on user behaviour and assume that the device is totally predictable — no adaptive behaviour, no autonomous behaviour. They are based on

an important underlying idea: that users can usefully be considered as rational agents applying their knowledge in the interaction (Cohen *et al.*, 1991; Pollock, 1993; Butterworth & Blandford, 1999). However, they all demand too much fine-grained detail to be extended readily to situations where the device is intelligent in any meaningful way.

3.2.2. User errors are minimized A focus on learning can be viewed alternatively as a focus on how users acquire knowledge of the system they are working with. Probably the most important source of user errors is mistakes due to incorrect or incomplete knowledge of how the system works or what state it is currently in. Some ways of thinking about user knowledge and misconceptions have been discussed above. Reason (1990) distinguishes between two types of errors: mistakes and slips. Mistakes are attributable to user misconceptions. Slips are the errors that are made when a user knows what to do but does the wrong thing. Causes include keys or menu items being too close together or lapses in concentration. Slips represent breakdowns in step 4 of Norman's action cycle. Hollnagel (1998) proposes a way of thinking about errors in terms of phenotypes and genotypes — terms that are familiar within AI to those working on genetic algorithms (GA). Just as, in work on GA, the phenotype refers to physical processes, so Hollnagel's phenotypes are concerned with the way an error manifests itself in interaction. Similarly, in the context of GA a genotype relates to the underlying encoding, so Hollnagel's genotypes are concerned with the underlying causes of an error. This might be to do with the way the system fails to support human memory at a time of high workload (Byrne & Bovair, 1997) or the way users' attention is — or is not — drawn to important information. An understanding of human properties such as working memory limitations and how people direct their attention, and how to design to minimize errors, applies equally to traditional and intelligent systems.

3.2.3. User's conceptual model fits well with that implemented within the device We have already discussed how users develop an understanding of the device, and how the designer can influence the user's understanding of device concepts through appropriate interface design. However, we have also discussed the idea that the device is supporting the user performing some domain tasks, and that there should be a good conceptual fit between the domain and the device. For example, discussing the design of electronic diaries, Cooper (1999) argues that there are two types of time-based information — deadlines and ongoing processes — and that the appointments that are implemented in all existing diary systems are inappropriate for both of these categories. In particular, 'appointments' are inadequate for representing ongoing processes. Real appointments have start times but rarely have pre-determined end

times, but diary systems force every appointment to have one. In addition, real appointments may notionally overlap, but many diary programs do not permit this; neither do they allow an appointment to have a start time and a preparation time (e.g. travelling time to get there). These are some of the ways in which electronic diaries have a poor conceptual fit to the tasks they are intended to support, and users of such tools need to find ways to express their requirements to their diaries, and to develop 'work arounds' to make them usable. Some approaches to task analysis, such as TKS (Johnson, 1992), aim to capture and describe the users' knowledge about domain objects, but this tends to capture the generalities and miss the detail — an example of a knowledge acquisition difficulty. Various approaches to capturing the detail and assessing the 'degree of fit' between domain and device concepts have been investigated, including ETIT (Moran, 1983), yoked state spaces (Payne *et al.*, 1990) and cognitive complexity theory (Kieras & Polson, 1985), but these have tended to prove difficult to work with.

This idea of conceptual fit has been addressed, if indirectly, in some work on tutoring systems, where the concern is to help the student acquire an appropriate conceptual model of the domain being studied. One early example was the adaptation of a medical expert system, MYCIN, by the addition of tutoring principles to create GUIDON (Clancey, 1987); despite careful design, it was found to be a poor tutor because the rules that supported expert diagnosis are not expressed in a way that matches the human expert's diagnosis strategy — much less that of a trainee who has yet to develop domain-specific expertise. Other systems developed at around the same time, such as STEAMER (Hollan *et al.*, 1984) and RBT (Woolf *et al.*, 1987), explicitly aimed to help the student acquire a mental model of the equipment being taught about in order to help in operating and troubleshooting that equipment, and were found to be more educationally effective. More recent tutoring systems have continued to focus attention on the representation of the domain material being taught through the interface, recognizing the importance of 'conceptual fit'.

3.2.4. User's device task structure maps naturally onto domain task structure As well as capturing essential domain concepts, it is generally important that task structures map naturally from domain to device. Domain task analysis — identifying domain task structures and expressing them in terms of hierarchical plans and procedures — has been widely used for many years (e.g. Annett & Duncan, 1967) and is well established as a means of defining device task procedures for traditional software design. While this works well for tasks with a strong procedural element, it is less effective for tasks that are more ill-structured, where the procedural element is less important than the conceptual. Similarly, at first sight this would appear to be unimportant for interactions between a user and an

intelligent system, but then the question might be: what *does* determine how natural a particular task structure — or interaction sequence — with such a system is? The flow of the interaction should make sense to all participants at all times. Put another way: the dialogue should remain coherent. Whereas for a task-oriented dialogue coherence is achieved indirectly by getting the task structure right, for other systems it is necessary to address coherence directly.

3.2.5. Discussion At the level of the interaction, the way traditional HCI has addressed the challenge of usability has been to assume that the computer system is a dumb tool and to analyse requirements from that perspective. Winograd and Flores (1986) draw on Heidegger's analogy of a hammer to discuss the idea that computer systems should be ready-to-hand. They argue that, just as a carpenter is generally unaware of the hammer until something goes wrong, such as hitting a thumb, so the user of a computer system should be almost unaware of it as they go about performing their task. So, for example, a word processor is supporting users as they create a document, and it should be 'invisible': users should not be concerned with its internals. Similarly, an information system, such as a set of documents on the Web, should be traversable in a way that allows users to focus their attention on the content rather than the navigation mechanism. Such applications may incorporate AI technologies (e.g. heuristics for proposing alternative words in a spell checker, or intelligent agents within a Web search engine), but that should not matter to the user. For such systems, the HCI theories and techniques discussed above generally apply. In particular, the goodness of fit between domain and device and the learnability of the device are central.

For other intelligent systems, whose behaviour is adaptive or otherwise unpredictable to the user, we need new ways of thinking about the problem. One approach is to reformulate the learnability requirement: that the user and computer system should seek to develop and maintain 'common ground' (Clark & Brennan, 1991). From this perspective, learnability is concerned with how easily the user can develop a shared understanding with the system, and conceptual fit is concerned with how well the computer system fits the user's starting point. Then dialogue coherence is a requirement for achieving and maintaining that common ground. Dialogue coherence and shared understanding are discussed in more detail below.

3.3. Context and organizational environment

While the requirement that a new system should fit well within the organizational context and working practices is an obvious one, it is one that has received relatively little attention within either the AI or the HCI community.

We start with a simple example of a study of organizational fit: diary management tools. Blandford and Green (in

press) found that the most important factors determining acceptance of a particular tool were not concerned with interface or system design details but with how the tool fitted in with the user's working practices — so, for example, people whose time is occupied by a large number of meetings value a shared diary system so that meetings can be scheduled relatively easily, and are prepared to pay the price for that in terms of diary maintenance time and making their diary information accessible to other users of the same diary system. People whose activities are more 'solo' are less keen to use such a system. Similarly, people who travel regularly require diaries that are easily ported, and placed anywhere, whereas people whose work takes place in one office will tend to opt for a desk diary or a computer-based one. Although usability is perceived as important, whether a diary is 'intelligent' or not is immaterial: decision criteria are based more on intrinsic design features such as portability, ease of access, ease of sharing information, and even social cachet, than on details of the implementation.

Heathfield (1999), working on medical expert systems for use in hospitals, found that social and organizational considerations strongly influenced uptake. She discusses the need for organizational acceptance of a tool before it will have an impact. The same point is made — rather more cautiously — by Fox *et al.* (1998), who argue that decision support and guideline systems need to be subjected to controlled clinical trials prior to adoption, so that they are accepted and trusted within the organization.

At the level of organization and context, the determinants of acceptability and acceptance are broadly similar for all types of system — or at least, the important differences are not based on whether or not the system is 'intelligent'.

3. 4. Discussion

We have argued that at the lowest level (implementation of technologies and details of interface design) and the highest level (social and organizational concerns), the issues facing usability and acceptance of intelligent systems are indistinguishable from those for traditional systems — that the important differences lie at the level of the interaction. We have sought an alternative way of expressing the core requirements on the interaction between people and systems regardless of the type of system — it is, after all, the same human beings using the different kinds of systems — and have argued that traditional HCI approaches have made assumptions about features of the technology being designed that do not translate well to intelligent systems.

4. Turning the question round: what do intelligent systems need?

The question then becomes: how can usability requirements be better expressed to accommodate the features of intelli-

gent systems? We have discussed some of the simpler examples — notably the extremes, where the intelligence is completely hidden from the user, and where the user cannot work without understanding the underlying technology. The greatest challenges reside in the central area — where the user is aware of, and has to work with, the intelligence of the system, but is not expected to have a deep understanding of the underlying implementation or AI theory. Hollnagel (1998, p. 71) argues that 'complex man-machine combinations are best conceptualised in terms of a joint cognitive system'. This is consistent with the view of distributed cognition, discussed above, that individuals do not work in isolation, but have an interdependent relationship with both people and other resources in their work environment. We can consider the relationships between people and devices in terms of the roles each plays within the 'joint cognitive system'; people adapt their roles frequently and fluidly according to the demands of the situation, but within limits they expect systems to behave in ways that could broadly be described as cooperative and comprehensible.

4. 1. Requirements on intelligent computer systems

The main requirements identified from an HCI perspective were discussed above. These are essentially requirements on any system; however, they are expressed in more general terms than is common for traditional systems.

1. The system should be learnable. The user should be able to work out at every stage within the interaction what next steps are possible and, of those, which might be desirable.
2. If viewed as a 'joint cognitive system' or a distributed system, the interactive system, including users and computer systems, should function effectively as a whole.
3. The interaction should be natural to the user. This might mean that the device implements an appropriate task structure or that it allows the user to do next whatever seems most appropriate to that user. A more general requirement is that the interaction remains coherent — that it satisfies natural constraints on the dialogue.
4. There should be a good fit between the domain problem as conceptualized by the user and the device representation. In other words, the user and computer should easily be able to establish and maintain 'common ground' — a topic that we discuss under the heading of 'ontological fit'.

If a system supports a natural, coherent dialogue, and if it is easy to establish and maintain common ground, then the system should be easy to learn. Therefore, we discuss interaction requirements under three headings: demands on the computer system within a 'joint cognitive system', dialogue coherence and ontological fit.

4. 2. Joint cognitive system

Figure 1 presented a view of an interactive system as a collection of interacting agents and objects. Many intelligent systems take the same role as a traditional computer system within such a network. However, there is another architecture that is common, particularly in the fields of intelligent agents and tutoring systems (ITSs), where the two agents at the centre of Figure 1 are replaced by three, as illustrated in Figure 3. Here the solid arrows indicate that agents may communicate directly, and shadowed arrows indicate that one agent can observe the actions of the other.

Within work on intelligent agents, the 'environment/application' is typically the application being used, and the role of the agent is often as advisor. Within work on tutoring systems, the 'environment' is more commonly a representation of the domain the user is intended to learn about, and the agent is a tutor, who may take on a particular role within the interaction, e.g. coach (Burton & Brown, 1979), co-learner (Chan & Baskin, 1990), collaborative problem solver (Blandford, 1994) or guide (Johnson *et al.*, 2000).

Such an architecture clearly raises new issues. For example, how is turn-taking managed? How does each agent remain appropriately aware of the activities of the others? How do conflicts between agents get resolved? System developers have tended to make local decisions that work in the particular setting they are investigating, but to date there is little applicable theory in this area.

To push the question of responsibility and authority a little further, we take a simple example: an intelligent electronic diary. Mueller (2000) describes a novel diary implementation, embodying AI techniques. The diary alerts the user if an appointment seems inappropriate; the example Mueller presents is that of booking a table at a steak house for a meal with a vegetarian. It also implements some intelligent defaults, e.g. that a meal is likely to last two hours, so a 'meal' entry in the diary is given a default length of two hours. The first of these features (the alert) is likely to be helpful; some users may not like such a feature, but it is unlikely to cause a scheduling disaster as the user retains control over the booking. However, one of the findings in the study reported by Blandford and Green

(in press) was that defaults such as appointment length can lead to scheduling difficulties. At the time of making the entry a user often does not even think about how long an appointment is likely to be, so does not consider the end time; but at a later date — e.g. when scheduling another appointment to follow the existing one — the user works with the explicit information in the diary and does not question it, trusting that information to be accurate. By implementing such a default, whether 'intelligent' or not, the computer system has effectively taken responsibility for something about which it has too little information, and the user, in relying on the computer, has inadvertently relinquished control.

This issue of roles and responsibilities has been recognized in some safety-critical application areas; e.g. Fox *et al.* (1998) discuss the importance of people taking responsibility for the decisions made, even when an intelligent system is used to support that decision, in the domains of aircraft maintenance and clinical decision-making. Billings (1997), in the context of aviation automation, presents a series of principles for the design and introduction of automation — whether intelligent or otherwise. One requirement is that the human operator must remain in command, and that therefore the operator must remain involved and always be appropriately informed, particularly about automated system behaviour. In addition, automated systems must be predictable, and each agent in the intelligent human-machine system must have knowledge of the intentions and actions of the other agents. Such principles are likely to apply in almost all domains, not just aviation, but clearly are not universal; e.g. network agents and remote monitoring agents may have delegated authority for a time, but will be required to transfer that authority back to a human operator under certain conditions. There is a broad class of issues that determine the success or otherwise of joint cognitive systems, which are only just beginning to be understood.

4. 3. Understanding dialogue structure

The second issue identified above was that of naturalness of dialogue. Within HCI, the term 'dialogue' is widely used to refer to any form of interaction between user and computer, regardless of whether or not it has any linguistic component, or any of the properties widely associated with human-human dialogue. Within other disciplines such as linguistics, the properties and structure of human-human dialogue have been extensively studied. For example, Grice (1975) proposes a set of maxims for the conduct of satisfactory dialogue that can be summarized overall as observing that each contribution to the dialogue is necessary, sufficient and relevant (so if some contribution does not instantly appear to satisfy these, the hearer will usually make inferences or reinterpret the utterance so that it does satisfy these conditions). While such maxims are stated in

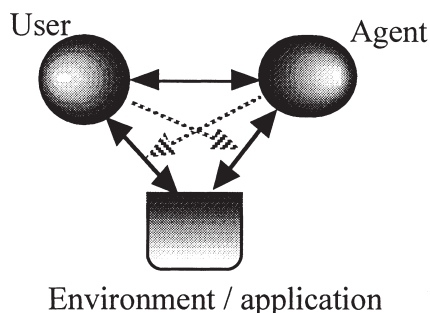


Figure 3: A system of agents.

too general a form to guide design, they provide a broad set of requirements on interaction.

Grosz and Sidner (1986) distinguish three aspects of discourse — attentional, intentional and structural. They define attention as being the topic, or focus, of the dialogue at a given instant. Intention refers to the purpose of the discourse segment, and structure to the grammatical structure of the discourse. They present a stack structure for the topic, such that the active topic is the one at the top of the stack, while other open topics are also in the stack. Topics are removed from the stack when they are closed. This work is purely descriptive, but serves to identify some constraints within which any dialogue participant has to operate if its utterances are to be recognizably coherent. In particular, as well as maintaining its version of the ‘shared model of the dialogue’, a dialogue participant has to keep a record of active, open and closed topics. However, coherence does not depend solely on the focus of the conversation, any more than it does solely on the function. Each utterance should be clearly relevant in the current context, so as to be easily integrated with the hearer’s model of the dialogue. The challenge is to design systems that can participate in coherent dialogue, and to evaluate the coherence of existing dialogue designs.

Little work has been done in HCI on ensuring that the interaction remains coherent, because this has not been a core issue in traditional systems: if the task structure of tools is satisfactory then the dialogue structure is likely to be adequate too. Similarly, for systems with which the user determines the task structure, the user also determines the ‘topic shifts’. However, there are counter-examples. For example, for many personal computers, the action of trying to copy a file to a (too full) disk typically goes as shown in Table 3 (U is the user and S the system). Arguably, if the system were designed to conform better to the principles of

natural dialogue, the final user action would be unnecessary because the system would already have the information that this was what the user intended. The topic was not closed at utterance S2, but put on the stack as an open (but not active) topic, and should naturally be reactivated when the intervening topic was closed.

While this example is fairly trivial, it illustrates an important point: that even a simple understanding of dialogue structures could be used to help design more natural dialogues, even for traditional systems. This becomes much more important for intelligent systems (e.g. Pilkington, 1992). However, many traditional expert systems violate this grossly, asking questions that would appear to most users to be completely random, as a direct result of the implementation of the underlying rule base.

Sidner and Coffman (1999) propose a vision for the future: that ‘The coming interface is one in which the user collaborates with the computer. The computer understands what the user is doing, can take part in those activities and is able to respond conversationally to the user’s activities.’ While many would challenge the realizability, or even desirability, of this vision, initial work in this direction is reported by Rich and Sidner (1997), who describe a collaborative interface agent that works with the user on a simple task — in this case, scheduling a flight. Both user and agent have equal access to the application domain: they can each interact with the application, each observe the other’s actions, and communicate directly with each other. The agent implements the dialogue principles identified in the earlier work of Grosz and Sidner (1986) and others, and therefore is able to maintain a record of shared plans and intentions and the current and other open topics, so that the overall dialogue can be appropriately structured to remain coherent to both participants. A similar approach was taken by Blandford (1993), in the design and implementation of a tutorial agent that developed shared plans with the student on how to solve a simple decision problem and then discussed the decision criteria with the student, maintaining a simple ‘stack’ of open topics within the dialogue.

While dialogue systems such as those described have not yet advanced beyond the ‘proof of concept’ stage of development, they serve to illustrate that it is possible to take a theory of what makes dialogue coherent and purposeful as a basis for design. However, to be more widely applicable in design and evaluation of interactive systems such a theory needs to be expressed appropriately for more general application. In addition, if the work is to be extended to multi-agent systems, then corresponding theories of turn-taking and interruptions also need to be developed and expressed appropriately.

4. 4. Ontological fit

The third issue identified above was that of conceptual fit. Authors including Winograd and Flores (1986) and Dreyfus

Table 3

	<i>Action</i>	<i>Meaning</i>
U1	Drag file icon over disk icon	Copy this file to this disk
S2	‘Disk full. To copy that file you need NK more space’	There is insufficient space on the disk
U3	Open disk, select files and drag them to trash can	Delete these files [to make space]
S4	Files disappear from ‘disk’ window	Done that
U5	Drag original file icon over disk icon	Now copy this file to this disk...

(1992) argue persuasively that computer systems will never be intelligent, in the rich sense of being able to act without an explicit representation on which to act. Whether or not this is the case, the current need of computer systems to work with explicit representations demands that users need to comprehend and adapt to those representations, to establish the common ground that is necessary for effective communication.

We have already discussed the idea that the agents (users and computer systems) within an interactive system need to establish 'common ground'. In an AI context, this idea appears in work on ontologies, where an ontology is defined by Gruber (1993) as 'a specification of a conceptualisation'. In particular, Gruber presents ontologies as enabling knowledge to be shared with and between agents without them needing to share a global theory or to have the same architectures. Guarino (1997) presents ontologies as being 'task-independent knowledge bases' — another attractive aspect when we consider that most computer systems are designed to support a wide variety of tasks, and that many of those tasks are ill-structured (Simon, 1973).

What does it mean to apply this idea of ontologies to HCI? We can relate it back to the notion discussed earlier of conceptual fit. If we can lay out the ontology with which a user is working and that represented within the computer system, then we have a starting point for reasoning about the possibility of establishing adequate common ground. In terms of learning, if the user has to be aware of many novel concepts implemented within the computer system, then that user is likely to experience difficulty learning to use the system. Conversely, if there are domain concepts that are centrally important to the user, that are not explicitly represented within the computer, then the user is going to have to find circuitous ways of expressing those ideas to the machine. One approach to analysing the quality of fit between users' and device ontologies is currently being explored in work on ontological sketch modelling (OSM) (Blandford & Green, 1997). Traditional empirical techniques (interviews, questionnaires, observation) are used to collect data on how users think about their domain tasks — which are then modelled in terms of entities and attributes that are important to users, and the known actions that change the state. In parallel, computer system descriptions (such as specifications, manuals or existing running systems) are used to create a device ontology of the same form. These descriptions can be combined to support reasoning about quality of fit. OSM has so far only been applied to traditional systems; it is one possible approach to assessing how easy it is to establish 'common ground'. There is scope for new techniques that support reasoning about how domain concepts should be represented within the computer system, and how they can be manipulated in ways that seem natural to users, while also presenting device-specific concepts to users in ways that make them

easy to assimilate and manipulate. These challenges apply to all systems, not just intelligent ones.

5. Conclusions

Inevitably, given the possible scope of the topic, the review presented here is selective, and some complex ideas and arguments have been treated in a simplified way. However, we have identified some of the important reasons for the limited interaction between the HCI and AI communities. In particular, the two communities have different agendas: much of the AI community is more concerned with what is possible than with what is usable, and much of the HCI community focuses on existing systems for which the system can be understood, though the interaction may present challenges to understanding. Each community is tackling difficult issues without taking the concerns of the other into account. This point is graphically illustrated by Partridge and Hussain (1995, p. 155), in their work on knowledge-based information systems (KBISs), who propose that 'interfaces are the last modules of a KBIS that need to be designed'. Given the difficulties of developing reliable and trustworthy KBISs, this may appear a reasonable position to take; however, it subordinates the needs of the end user to those of the developer and, in the longer term, reduces the uptake and acceptance of such systems.

We have identified some tools and techniques that can be adopted with little adaptation from HCI to AI for the development of usable systems. In particular, there is a substantial body of work on the details of interaction and computer system design, and a smaller body of work on understanding use in context, that can be taken directly from work on HCI for traditional systems and applied without modification to the design, implementation and evaluation of intelligent systems. However, at the level of interactions, the repertoire of reusable techniques is too small, and there is a shortage of relevantly expressed theories and methods for dealing with the trickiest issues in the development of usable intelligent systems.

In this review, we have also identified some points of commonality between AI and HCI, e.g. in the work of Hollnagel on phenotypes and genotypes, and the approaches to cognitive modelling that derive from AI theories. These approaches, or expressions of theory, may provide a starting point for useful HCI for intelligent systems, but they do not offer a solution to the challenge of developing theory-based HCI techniques that address the most difficult challenges in the design of usable intelligent systems. More optimistically, there is existing work within the AI community that is a promising starting point for a new research agenda for a theory for intelligent interaction design and evaluation. Three essential components of such a theory are concerned with structure, process and multi-agent coordination. In principle, appropriately expressed theory should apply equally to traditional and intelligent interactions,

though local theories that deal with particular kinds of interactions will also have their place.

The structure of an interaction can be thought of in terms of dialogue, or multi-party communication. An important requirement on the interaction is that it should remain coherent, being natural to all participants, minimizing the likelihood of breakdowns and making conversational repairs easy. Existing theories of dialogue appear to be a good starting point for developing such theory. Clearly, a general theory will also need to accommodate many of the insights from task analysis and cognitive modelling that have become established within HCI, but it will also need to go beyond them. Local theories that apply to particular kinds of intelligent systems will need to encapsulate core aspects that determine coherence, while abstracting away from others.

Maintenance of adequate shared understanding, or 'common ground', is essential to effective interaction. We have proposed that work on ontologies can contribute to a better understanding of this aspect of interaction. For multi-agent systems, there is a need for a representation that focuses on coordination and the development and maintenance of necessary shared understanding, so that each agent can fulfil their responsibilities within the interaction. What has historically been expressed as requirements on a tool needs to be expressed as requirements on a 'joint cognitive system', with the understanding that the user components of that system are difficult to redesign.

We are a long way from any grand universal theory of interaction with intelligent systems; in the short term, at least, focused theories that address particular interaction challenges are needed if intelligent systems are to achieve their potential in practical applications.

Acknowledgements

I am grateful to Paul Cairns, Ian Mitchell, Gordon Rugg and William Wong for constructive criticisms of drafts of this paper.

References

- ANNETT, J. and K.D. DUNCAN (1967) Task analysis and training design, *Occupational Psychology*, **41**, 211–221.
- AVISON, D.E. and G. FITZGERALD (1995) *Information Systems Development: Methodologies, Tools and Techniques*, 2nd edn, New York: McGraw-Hill.
- BERTIN, A., F. BUCIOL and A. STEFANINI (1998) Towards industrial application of intelligent training systems, *Expert Systems*, **15**, 1–21.
- BEYER, H. and K. HOLTZBLATT (1998) *Contextual Design*, San Mateo, CA: Morgan Kaufmann.
- BILLINGS, C.E. (1997) *Aviation Automation*, Hillsdale, NJ: Lawrence Erlbaum.
- BLANDFORD, A.E. (1993) An agent-theoretic approach to computer participation in dialogue, *International Journal of Man-Machine Studies*, **39** (6), 965–998.
- BLANDFORD, A.E. (1994) Teaching through collaborative problem solving, *Journal of Artificial Intelligence in Education*, **5**, 51–84.
- BLANDFORD, A. and T. GREEN (1997) OSM: an ontology-based approach to usability evaluation, in *Proceedings of the International Workshop on Representations in Interactive Software Development*. Queen Mary and Westfield College, University of London, 82–91.
- BLANDFORD, A. and T. GREEN (in press) Group and individual time management tools: what you get is not what you need, *Personal and Ubiquitous Computing*, forthcoming.
- BLANDFORD, A.E. and R.M. YOUNG (1996) Specifying user knowledge for the design of interactive systems, *Software Engineering Journal*, **11** (6), 323–333.
- BURTON, R.R. and J.S. BROWN (1979) An investigation of computer coaching for informal learning activities, *International Journal of Man-Machine Studies*, **11**, 5–24.
- BUTTERWORTH, R.J. and A.E. BLANDFORD (1999) The principle of rationality and models of highly interactive systems, in *Human-Computer Interaction INTERACT '99*, M.A. Sasse and C. Johnson (eds), Amsterdam: IOS Press, 417–424.
- BYRNE, M.D. and S. BOVAIR (1997) A working memory model of a common procedural error, *Cognitive Science*, **21** (1), 31–61.
- CARD, S.K., T.P. MORAN and A. NEWELL (1983) *The Psychology of Human-Computer Interaction*, Hillsdale, NJ: Lawrence Erlbaum.
- CAWSEY, A.J., R.B. JONES and J. PEARSON (2000) The evaluation of a personalised health information system for patients with cancer, *User Modeling and User-Adapted Interaction*, **10**, 47–72.
- CHAN, T.W. and A.B. BASKIN (1990) Learning companion systems, in *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education*, C. Frasson and G. Gauthier (eds), New Jersey: Ablex, Ch. 1.
- CLANCEY, W.J. (1987) *Knowledge-Based Tutoring: The GUIDON Program*, Cambridge, MA: MIT Press.
- CLARK, H.H. and S.E. BRENNAN (1991) Grounding in communication, in *Perspectives on Socially Shared Cognition*, L.B. Resnick, J. Levine and S.D. Behrend (eds), Washington, DC: American Psychological Association, 127–149.
- COCKBURN, A. and S. JONES (1996) Which way now? Analysing and easing inadequacies in www navigation, *International Journal of Human-Computer Studies*, **45**, 105–129.
- COHEN, P., J. MORGAN and M. POLLACK (eds) (1991) *Intentions in Communication*, Cambridge, MA: MIT Press.
- COOPER, A. (1999) *The Inmates are Running the Asylum*, Indianapolis, IN: Sams Publishing.
- DAHLBÄCK, N., A. JÖNSSON and L. AHRENBORG (1993) Wizard of Oz studies — why and how, in *Proceedings of the Intelligent User Interfaces Workshop*, New York: ACM, 193–200.
- DIX, A., J. FINLAY, G. ABOWD and R. BEALE (1998) *Human-Computer Interaction*, 2nd edn, Englewood Cliffs, NJ: Prentice Hall International.
- DREYFUS, H.L. (1992) *What Computers Still Can't Do*, Cambridge, MA: MIT Press.
- DU BOULAY, B., T. O'SHEA and J. MONK (1981) The black box inside the glass box: presenting computing concepts to novices, *International Journal of Man-Machine Studies*, **14**, 237–249.
- ERICSSON, K.A. and H.A. SIMON (1984) *Protocol Analysis: Verbal Reports as Data*, Cambridge, MA: MIT Press.
- FIKES, R. and N. NILSSON (1971) STRIPS: a new approach to the application of theorem proving to problem solving, *Artificial Intelligence*, **2**, 189–208.
- FOX, J., N. JOHNS and A. RAHMANZADEH (1998) Disseminating medical knowledge: the PROforma approach, *Artificial Intelligence in Medicine*, **14**, 157–181.

- GOULD, J.D. (1988) How to design usable systems, in *Handbook of Human-Computer Interaction*, M. Helander (ed.), Amsterdam: Elsevier, 757-789.
- GRAY, W., B. JOHN and M. ATWOOD (1993) Project Ernestine: validating a GOMS analysis for predicting and explaining real-world task performance, *Human-Computer Interaction*, **8**, 237-309.
- GRICE, H.P. (1975) Logic and conversation. In *Syntax and Semantics 3*, P. Cole and J.L. Morgan (eds), New York: Academic Press, 41-58.
- GROSZ, B.J. and C.L. SIDNER (1986) Attention, intention and the structure of discourse, *Computational Linguistics*, **12**, 175-204.
- GRUBER, T.R. (1993) A translation approach to portable ontologies, *Knowledge Acquisition*, **5**, 199-220.
- GUARINO, N. (1997) Understanding, building and using ontologies, *International Journal of Human-Computer Studies*, **46**, 293-310.
- HACKOS, J. and J. REDISH (1998) *User and Task Analysis for Interface Design*, New York: Wiley.
- HALL, A. (1990) Seven myths of formal methods, *IEEE Software*, September, 11-19.
- HEATHFIELD, H. (1999) The rise and 'fall' of expert systems in medicine, *Expert Systems*, **16**, 183-188.
- HOLLAN, J.D., E.L. HUTCHINS and L.M. WEITZMAN (1984) STEAMER: an interactive, inspectable simulation-based training system, *AI Magazine*, **5** (2), 15-27.
- HOLLNAGEL, E. (1998) *Cognitive Reliability and Error Analysis Method (CREAM)*, Oxford: Elsevier Science.
- HOLLNAGEL, E. and D. WOODS (1983) Cognitive systems engineering: new wine in new bottles, *International Journal of Man-Machine Studies*, **18**, 583-600.
- HUTCHINS, E. and T. KLAUSEN (1991) Distributed cognition in an airline cockpit, in *Cognition and Communication at Work*, Y. Engeström and D. Middleton (eds), Cambridge: Cambridge University Press.
- JAMESON, A., R. SCHÄFER, T. WEIS, A. BERTHOLD and T. WEYRATH (1999) Making systems sensitive to the user's time and working memory constraints, in *Proc. IUI 1999*, New York: ACM, 79-86.
- JOHN, B. and D.E. KIERAS (1996) Using GOMS for user interface design and evaluation: which technique?, *ACM ToCHI*, New York: ACM, 1-30.
- JOHNSON, P. (1992) *Human-Computer Interaction: Psychology, Task Analysis and Software Engineering*, London: McGraw-Hill.
- JOHNSON, W.L., J.W. RICKEL and J.C. LESTER (2000) Animated pedagogical agents: face-to-face interaction in interactive learning environments, *International Journal of Artificial Intelligence and Education*, **11**, 47-78.
- JOHNSON-LAIRD, P.N. (1981) *Mental Models*, Cambridge: Cambridge University Press.
- KIERAS, D. and P. POLSON (1985) An approach to the formal analysis of user complexity, *International Journal of Man-Machine Studies*, **22**, 356-394.
- KOHONEN, T. (1984) *Self Organisation and Associative Memory*, Berlin: Springer.
- LANDAUER, T.K. (1995) *The Trouble with Computers: Usefulness, Usability and Productivity*, Cambridge, MA: MIT Press.
- LAUREL, B. (1990) Interface agents: metaphors with character, in *The Art of Human-Computer Interface Design*, B. Laurel (ed.), Reading, MA: Addison-Wesley.
- LEE, Y.-H. and M.W. EVENS (1998) Natural language interface for an expert system, *Expert Systems*, **15**, 233-239.
- LESTER, J.C., S.A. CONVERSE, S.E. KAHLER, S.T. BARLOW, B.A. STONE and R.S. BHOGAL (1997) The persona effect: affective impact of animated pedagogical agents, *Proc. ACM CHI '97*, New York: ACM, 359-366.
- LIEBERMAN, H. (1997) Autonomous interface agents, *Proc. ACM CHI '97*, New York: ACM, 67-74.
- LOHSE, G.L. (2000) Usability and profits in the digital economy, in *Proc. HCI2000*, S. McDonald, Y. Waern and G. Cockton (eds), Berlin: Springer, 3-17.
- LUGER, G.F. and W.A. STUBBLEFIELD (1998) *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, 3rd edn, Reading, MA: Addison-Wesley.
- MAY, J. and P. BARNARD (1995) Cinematography and interface design, in *Human-Computer Interaction: Interact '95*, K. Nordby, P. Helmersen, D.J. Gilmore and S. Arnesen (eds), London: Chapman and Hall, 26-31.
- MAY, J., P. BARNARD and A. BLANDFORD (1993) Using structural descriptions of interfaces to automate the modelling of user cognition, *User Modelling and User-Adapted Interaction*, **3** (1), 27-64.
- MORAN, T.P. (1983) Getting into a system: external-internal task mapping analysis, *Proc. ACM CHI '83*, New York: ACM, 45-49.
- MUELLER, E.T. (2000) A calendar with common sense, *Proc. IUI 2000*, New York: ACM, 198-201.
- MUMFORD, E. (1995) *Effective Requirements Analysis and Systems Design: The ETHICS Method*, Basingstoke: Macmillan.
- NIELSEN, J. (1994) Heuristic evaluation, in *Usability Inspection Methods*, J. Nielsen and R. Mack (eds), New York: Wiley, 25-62.
- NORMAN, D. (1986) Cognitive engineering, in *User Centered System Design*, D.A. Norman and J.W. Draper (eds), Hillsdale, NJ: Lawrence Erlbaum, 31-62.
- PARTRIDGE, D. and K.M. HUSSAIN (1995) *Knowledge Based Information Systems*, London: McGraw-Hill.
- PARTRIDGE, D. and Y. WILKS (1987) Does AI have a methodology which is different from software engineering?, *Artificial Intelligence Review*, **1**, 111-121.
- PAYNE, S.J., H.R. SQUIBB and A. HOWES (1990) The nature of device models: the yoked state space hypothesis, and some experiments with text editors, *Human-Computer Interaction*, **5**, 415-444.
- PILKINGTON, R.M. (1992) *Intelligent Help: Communicating with Knowledge-Based Systems*, London: Paul Chapman.
- POLLOCK, J. (1993) The phylogeny of rationality, *Cognitive Science*, **17**, 563-588.
- PREECE, J., Y. ROGERS, H. SHARP, D. BENYON, S. HOLLAND and T. CAREY (1994) *Human-Computer Interaction*, Harlow: Addison-Wesley.
- REASON, J. (1990) *Human Error*, Cambridge: Cambridge University Press.
- RICH, C. and C.L. SIDNER (1997) Segmented interaction history in a collaborative interface agent, in *Proc. IUI '97*, New York: ACM.
- SELF, J. (1999) The defining characteristics of intelligent systems research: ITSs care, precisely, *International Journal of Artificial Intelligence and Education*, **10**, 350-364.
- SHNEIDERMAN, B. (1998) *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Reading, MA: Addison-Wesley.
- SIDNER, C. and D. COFFMAN (1999) Collaborative, spoken-language interface agents, *Proc. IUI 1999*, New York: ACM, 97.
- SIMON, H.A. (1973) The structure of ill-structured problems, *Artificial Intelligence*, **4**, 181-200.
- STRAUSS, A. and J. CORBIN (1998) *Basics of Qualitative Research*, Thousand Oaks, CA: Sage.

- WHARTON, C., J. RIEMAN, C. LEWIS and P. POLSON (1994) The cognitive walkthrough method: a practitioner's guide, in *Usability Inspection Methods*, J. Nielsen and R. Mack (eds), New York: Wiley, 105–140.
- WILSON, M., D. DUCE and D. SIMPSON (1989) Life cycles in software engineering and knowledge engineering: a comparative review, *Knowledge Engineering Review*, **4**, 189–204.
- WINOGRAD, T. and F. FLORES (1986) *Understanding Computers and Cognition*, Reading, MA: Addison-Wesley.
- WOOLF, B., D. BLEGEN, J. JANSEN and A. VERLOOP (1987) Teaching a complex industrial process, in *Artificial Intelligence and Education*, Vol. 1, R. Lawler and M. Yazdani (eds), Norwood, MA: Ablex.
- WRIGHT, P.C., R.E. FIELDS and M.D. HARRISON (2000) Analysing human–computer interaction as distributed cognition: the resources model, *Human–Computer Interaction Journal*, **15**, 1–41.
- XU, L.D. and L.X. LI (2000) A hybrid system applied to epidemic screening, *Expert Systems*, **17**, 81–89.
- YOUNG, R.M. (1981) The machine inside the machine: users' models of pocket calculators, *International Journal of Man–Machine Studies*, **15**, 51–85.

YOUNG, R.M., T.R.G. GREEN and T. SIMON (1989) Programmable user models for predictive evaluation of interface designs, in *Proceedings of CHI '89*, New York: ACM.

The author

Ann Blandford

Ann Blandford is Professor of Interaction Design at Middlesex University. She was awarded her doctorate in artificial intelligence and education from the Open University in 1991, and was Chair of AISB, the UK Society for the Study of Artificial Intelligence and the Simulation of Behaviour, 1997–99. She has published widely on artificial intelligence and education, design methods and usability evaluation, with a particular focus on theory-based approaches to evaluation.

Copyright of Expert Systems is the property of Wiley-Blackwell and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.