Ambient Intelligence
G. Riva, F. Vatalaro, F. Davide, , M. Alcañiz (Eds.)
IOS Press, 2005, http://www.ambientintelligence.org

9 Interactive Context-Aware Systems Interacting with Ambient Intelligence

Albrecht SCHMIDT

Abstract. Interactive systems have been the dominant computing paradigm over recent years. This paradigm is characterized by the fact that human user and the system communicate and interact explicitly using different modalities. However to come closer to visions of Ambient Intelligence, Calm Computing, Disappearing Computing, and Ubiquitous Computing new forms of interaction are required. Observing humans interacting with each other and new possibilities given by emerging technologies indicate that a new interaction model is needed. In this chapter we present the concept of implicit human computer interaction (iHCI) that takes the users context into account when creating new user interfaces for ambient intelligence. Beyond the model examples are given, application areas are described and basic implementation issues are discussed. Our research leads to a more general discussion on disappearing and invisible user interfaces. In the invisibility dilemma we explain that in many application areas there may be an inherent conflict. The transparent user interface and the added value gained by introducing technology are often opposing goals, especially combined with users that are creative in appropriating their tools.

Contents

9.1 Introduction

Context and context-awareness are central issues to ambient intelligence. The availability of context and the use of context in interactive applications offer new possibilities to tailor applications and systems "on-the-fly" to the current situation. However, context influences and often fundamentally changes interactive systems. In this chapter we first provide a brief introduction to interactive applications, traditional interaction paradigms, and the way humans interact with each other. Then the concept of implicit human computer interaction (iHCI) is introduced and illustrated by a discussion of examples and application domains. A basic implementation problem – whether to push or pull context – is addressed. In the final part of the chapter the invisibility dilemma is introduced. Here the basic problem of creating invisible and ambient user interfaces is addressed.

9.2 Interaction and Interactive Applications

The communication of information from computer systems to a human user and influencing the operation of the computer system by a human user is referred to as human-computer-interaction (HCI).

Interactive applications offer a timely bi-directional communication between the human user and the computer system. When using interactive applications the user and the system are in a direct dialog. This dialog is a sequence of communication events between the user and the system [1, chapter 3]. Interactive applications have evolved over the last 35 years, use different modalities, and are applied in various application areas. The distinctive property of interactive systems is that there is a direct and timely interaction between the user and the system. Non-interactive systems, such as batch processing of punch cards as used in the sixties or background processes in current systems do not allow a direct dialog between the user and the program.

Typical user interfaces (UIs) of interactive programs are text based (e.g. command line), graphical user interfaces, voice interfaces, gesture interfaces or a combination of those, often referred to as multimodal interfaces.

A characteristic feature of interactive systems is the response time, the time between the user interaction that is carried out and the response of the system [2], [3]. Most applications that are used on desktop systems in the home and office domain, such as text processor, spreadsheet, graphic tools, web browser, and games can be regarded as interactive programs.

Also the operating system itself and many programs that are running in the background often include interactive modules, mainly for configuration purpose.

Human computer interaction is not restricted to conventional desktop systems. As processing devices (e.g. logic circuits, DSPs, and microcontrollers) are included in many other interactive devices, such as VCRs, cameras, and mobile phones, *human-device interaction* becomes an important design criterion. Designing interaction and user interfaces for such systems has its distinctive challenges depending on the type of device.

Examples for UI consideration on PDAs are comprehensively analyzed in [4].

Design, development and implementation of interactive systems are extensively researched and for most modalities guidelines, approaches, methods, and tools are widely described and available. Commonly used approaches are graphical user interfaces (GUIs) that are build on event based interaction. The basic concept is to assign events to interactions carried out by the

user (e.g. pressing a button, dragging an icon). In the applications these events are linked to actions (e.g. calls of certain functions). For the development of applications using GUIs and user generated events development support is widely available at different levels in most current programming languages and development environments.

Interactive applications are not restricted to a single application, they can also be distributed. Here a standard method is to separate the UI from the processing component.

Applications implemented based on Web infrastructure are a typical example of this type of interactive applications. The visualization of the content and the immediate interaction is at the users system. However the response time of the server influences the interactive user experience.

9.2.1 Traditional and Explicit Human Computer Interaction

A key criterion of interactive applications is that they are used explicitly by the user. The basic procedure of a user initiated explicit interaction can be summarized by the following steps:

1. The user requests the system to carry out a certain action
2. The action is carried out by the computer, in modern interfaces providing feedback on this process
3. The system responds with an appropriate reply, which in some cases may be empty.

Consider the example of moving a file from one folder to another folder using a GUI.

The user drags the file from the source folder to the destination folder requesting by these means explicitly the move action (1). The system moves the file from one folder to the other providing progress visualization (2). After the interaction the GUI is presented with the file in the destination folder (3).

When observing an interaction that is initiated by the system, then the steps are preceded by a step where the system provides notification to the user. In certain cases reaction from the user is enforced (e.g. a system modal dialog box). In other cases it is up to the user whether or not to take action (e.g. ringing of a phone, email audio cue).

The interaction model “the execution-evaluation cycle” discussed by Norman [5] reflects a similar pattern, however 1) and 3) are subdivided into more detail.

This elementary interaction structure can be found in simple command line systems, in graphical direct manipulation interfaces [6], and also in systems using speech recognition and natural language processing. All these interfaces have in common that the user explicitly requests an action from the computer. However the way this request is formulated varies a lot, from cryptic but powerful text based commands with many parameters (e.g. in shells), manipulation of graphical objects in a GUI, and by spoken commands. The basic interaction of these communication processes is similar. The main difference between these modalities is the representations of objects and interactions.

There are different levels of abstraction that certain commands offer, the level of abstraction, however is widely independent of the modality used. With regard to usability and the time needed to learn how to operate a system significant differences are observed [6], [1, chapter 4].

In the following the group of conventional interactive systems as described above will be refereed to as system with explicit interaction, independent of their modality.

Observation: Explicit interaction contradicts the idea of invisible computing, disappearing interfaces, and ambient intelligence. New interaction paradigms are required to realize the vision of a Ubiquitous Computing environment which can offer natural interaction. It appears that explicit interaction – independent of the modality – is not sufficient to reach the goal.

Explicit interaction requires always a kind of dialog between the user and a particular system or computer the user is currently interacting with. This dialog brings the computer inevitably to the centre of the activity and the users focus is on the interface or on the interaction activity.

This form of interaction is obviously in contrast to the visions of calm and Ubiquitous Computing [7], [8]. Also the idea of a disappearing computer [9] and ambient intelligence is hard to imagine with explicit interaction only. The realization of these visions can only be achieved when parts of the interaction between the computer and the human are transparent and not explicit, as stated above.

9.2.2 Excurse: Interaction and Communication Between Humans

Interaction between humans is the most natural form of interaction human's use. This type of communication and interaction is highly complex and manifold. A complete model of this form of interaction seems at the moment impossible. Nevertheless analyzing key issues in interaction and communication between humans offers a starting point for a quest for new forms of interaction. In the following especially the influence of context will be central, and in particular three concepts: shared knowledge, communication error recovery, and surrounding situation.

Shared Knowledge

When observing communication and interaction between humans it is apparent that a common knowledge base is essential for understanding each other. The common knowledge is extensive and is usually not explicitly mentioned. A discrepancy in the shared knowledge often leads to communication problems as probably most people have experienced in everyday life, especially when travelling abroad. Any communication between humans takes a minimum common knowledge for granted. In most cases this minimum common knowledge however includes a complete world and language model, which however seems obvious but is very hard to grasp formally.

A search for modeling this knowledge, knowledge representation, and to make this knowledge accessible for machines has influenced many approaches in research in robotics and artificial intelligence. The expectation of humans towards other humans and to some extent also towards machines and computers is strongly influenced by the implicitly shared common knowledge.

Communication Errors and Recovery

Communication between humans is not at all error free. Many conversations include short term misunderstandings and ambiguities; however in a dialog these problems are resolved by the communication partners. Often ambiguities are rephrased and put into the conversation again to get clarity by reiteration of the issue. Similarly misunderstandings are often detected

by the monitoring the response of the communication partner. In case there is a misinterpretation issues are repeated and corrected.

When monitoring conversations it becomes apparent that efficient communication relays heavily on the ability to recognize communication errors and to resolve them. When building interactive systems that are invisible the ability to detect communication problems and to have ways to resolve it becomes crucial. In certain cases knowledge about the situation can provide the essential cues to solve the problem.

Situation and Context

Communication and interaction between humans happens always in a specific situation, a certain context, and in a particular environment.

When observing verbal communication it can be seen that the meaning of words, sentences, and also the communication behaviour, as well as the way the communication is carried out, is heavily influenced by the situation, context, and environment. The situation in which the communication takes place provides a common ground. This common ground generates implicit conventions, which influence and to some extent set the rules for interaction and also provide a key to decode the meaning of words and gestures. Single words have often many different meanings but the context and situation is the key to the "right" meaning. The behaviour related to the communication, e.g. initiating a communication, is also greatly dependent on the situation, and in particular the cultural conventions, roles of the participants, and communication goals. The type of conversation (e.g. formal or informal) is also defined by the situation.

In the field of natural language processing the situational knowledge is often reduced to the textual context. In [10] an analysis of this view on context and its role for understanding natural language is given. However, non-verbal communication, such as body language and gestures, is also essential for decoding spoken language. With body language and gestures information is shared in an implicit and subtle way which can be significant for the overall communication. A simple example is that humans recognize if their communication partner is in a hurry or not. Given this implicit knowledge the communication is most likely different for either case. The ability to learn and interpret implicit communication is a part of the social education and critical to be accepted as an appropriate communication partner.

Regarding applications and interaction processes with computers that are carried out in context, it seems natural that the context has a major influence on the interaction process.

Examples for relevant context information are:

- Verbal context (direct communication)
- Roles of communication partners
- Goals of the communication, goals of individuals
- Local environment (absolute, relative, types of environment, e.g. office or street)
- Social environment (e.g. who is there?)
- Physical and chemical environment.

Comparing the complex ways in which people interact to the way humans are operating machines, it becomes apparent that in general HCI does take the real world context of interaction and situation only very little into account. What humans expect when interacting

with other humans is dependent on the situation. We expect other people to act appropriate to a certain situation. However, as this is little regarded in current HCI most computers (and in a wider sense systems that include computer technology) do not react appropriately to a situation. This is easy to explain with the following example. Two people are in a conversation. A third person likes to remind one of them about a meeting that is going to take place in 10 minutes. Typically the person will wait for an appropriate pause in the communication and then interrupt and tell the person about the meeting. Also the level of detail will be appropriate to the situation. In contrast the calendar on a PDA will notify the user at a certain time with a certain level of detail independent of the circumstances.

These observations on the differences between interaction between humans, and current computer systems motivate the quest for new forms of human computer interaction.

9.3 The Concept of Implicit Human Computer Interaction (iHCI)

As explained above there are many things that influence the interaction between humans that are not contained in traditional "human computer interaction". The influence of situation, context, and environment offers a key to new ways of HCI. To come closer to the aim of creating interaction between humans and systems that is closer to natural interaction it becomes crucial to included implicit elements into the communication in addition to the explicit dialog that we already use.

The following definition characterizes the new paradigm of implicit human computer interaction (iHCI). In this chapter the focus is mainly on implicit input. However within the research also implicit output and the related concepts of ambient media were investigated.

The results are published in [11], [12], [13].

Definition: *Implicit Human-Computer Interaction (iHCI)*
iHCI is the interaction of a human with the environment and with artefacts which is aimed to accomplish a goal. Within this process the system acquires *implicit input* from the user and may present *implicit output* to the user.

Definition: *Implicit Input*
Implicit input are actions and behaviour of humans, which are done to achieve a goal and are not primarily regarded as interaction with a computer, but captured, recognized and interpret by a computer system as input.

Definition: *Implicit Output*
Output of a computer that is not directly related to an explicit input and which is seamlessly integrated with the environment and the task of the user.

The basic idea of implicit input is that the system can perceive the users interaction with the physical environment and also the overall situation in which an action takes place.

Based on the perception the system can anticipate the goals of the user to some extent and hence it may become possible to provide better support for the task the user is doing.

The basic claim is that Implicit Human Computer Interaction (iHCI) allows transparent usage of computer systems. This enables the user to concentrate on the task and allows centering the interaction in the physical environment rather then with the computer system.

A similar concept called "incidental interaction" is introduced in [14].

Realizing implicit input reliably as general concept appears at the current stage of research

close to impossible. A number of subtasks for realizing implicit input, such as recognition and interpretation of situations as well as general anticipation of user intension, are not solved yet.

However in restricted domains it is feasible, and as the following examples shows often simple or even trivial. The following examples are devices that are already used or easy to imagine. These systems incorporate the basic idea of iHCI without naming the paradigm explicitly.

9.3.1 Motivation and Examples of iHCI

A very simple example of a device that incorporates the basic concept of iHCI is an automatic outdoor lantern. Such lights are often found at the entrance of buildings.

Whenever a human comes close and it is dark the light switches automatically on. Two simple sensors (light level and PIR) are used to acquire the context. A simple electronic circuit detects the situation of interest. The situation is then hard-coded with an action (switching on the light for a certain period of time). The link between situation and action comes from the anticipation that the person wants light when moving towards the place. In this example the recognition of the situation, the interpretation, and the reaction is simple to describe and to implement.

Using additional sensors and communication technology the following scenarios can be easily implemented, some are commercially available. These examples motivate the starting point for iHCI, however most of them are currently still not widely used.

- The user drives into the driveway with her car. The car and the garage are equipped with communication units. The car communicates with the garage (e.g. a challenge response authentication protocol) and if the car has permission to enter the doors open automatically
- The heating/air condition control system of an office building has access diaries of the people working in the building. Office rooms are not heated/cooled when people work offsite or are away. Meeting rooms are heated/cooled in advance of scheduled meetings
- A garment that can measure pulse, skin temperature, and breathing combined with an outdoor location sensor and a communication unit can be used to monitor a users vital health signals. In case of a problem an emergency call can be issued.

In contrast the following examples for iHCI show that recognizing the situation as well as to reason about the user intension is non-trivial and often extremely hard. Even for relatively simple problem domains, such as light and device control in a home environment, this is difficult. One problem is to recognize situations reliably. A further problem, often an even more difficult one, is to assign user intensions to situations.

Consider the following example of a reading light and a TV. When the user is sitting in the arm chair reading a book the reading light should be on, when he shifts the attention towards the TV then it should be switched on. When the user takes again a book or a news paper and goes back to reading the TV should be switched off and the reading light should be on. The recognition of the situations seems feasible to some extent and also to link actions to it, however it is easy to construct cases where the system fails. E.g. the user watches TV and turns to TV-guide magazine. How should the system react? This also opens the question how transitions are made and how long situations have to last before they are taken into account.

9.3.2 Analyzing iHCI

Observing these examples and considering applications leads to the basic question of what the model for iHCI is. In particular the issue of how to link context to actions is a central concern.

In this section the basic principals on iHCI will be accessed which are then taken up by the model introduced later.

Analyzing applications and domains relevant to iHCI the following basic issues are central and have to be addressed in order to create such applications:

- *Perception as precondition*. To create applications that offer iHCI capabilities it is inevitable to provide the system with perception for context. This includes the domains of sensing, abstraction and representation
- Finding and analyzing *situations relevant for the application.* When applications are based on implicit interaction it becomes a central problem to find the situations that should have an effect on the behaviour of the system
- *Abstracting from situations* to context. Describing a situation is already an abstraction. To describe what should have an influence on applications classes of situations have to be selected which will influence the behaviour of an application
- *Linking context to behaviour*. To describe an iHCI applications classes of situations and in a more abstracted way contexts must be linked to actions carried out by the system.

Furthermore when considering the use and development of iHCI systems the following questions become imminent.

As it is often not possible to describe contexts, especially reflecting complex types of situations, in well defined sets the following question arises: How to represent fuzzy borders and dynamic thresholds?

When users interact with a system, interface stability is a critical issue. However, the concept of iHCI includes that without explicit user intervention changes are happening.

Two central questions come out of this issue:

- How to achieve a balance between stability and dynamic using concepts such as refractory periods and hysteresis?
- How to keep the user in charge of the interaction and not wondering about the actions taken by the system?

As implicit interaction is rarely the only form of interaction, it becomes important that it can be integrated with explicit interaction. How can implicit interaction be tied in with explicit interaction?

Implicit interaction is often ambiguous. Ways have to be found to deal with this issue.

Work in the area of ambiguity in interfaces is investigated in [15]. This puts the question: How to deal with ambiguities in iHCI?

9.3.3 The iHCI Model

To support the creation of systems that use implicit interaction it is important to provide a simple model that reflects this interaction paradigm. In Figure 9.1 an abstract model of implicit interaction is shown.

All actions carried out by a human are taking place in context – in a certain situation.

Usually interaction with our immediate environment is very intense (e.g. sitting on a chair, feet on the ground, garment on the body, moving books on the table, drinking from a glass, etc.) even if we don't recognized it to a great extent.

All contexts and situations are embedded in the world, but the perception of the world is dictated by the immediate context someone is in. Explicit user interaction with an application is embedded into the context of the user and is also a way of extending the context of the user, e.g. by having access to the network.

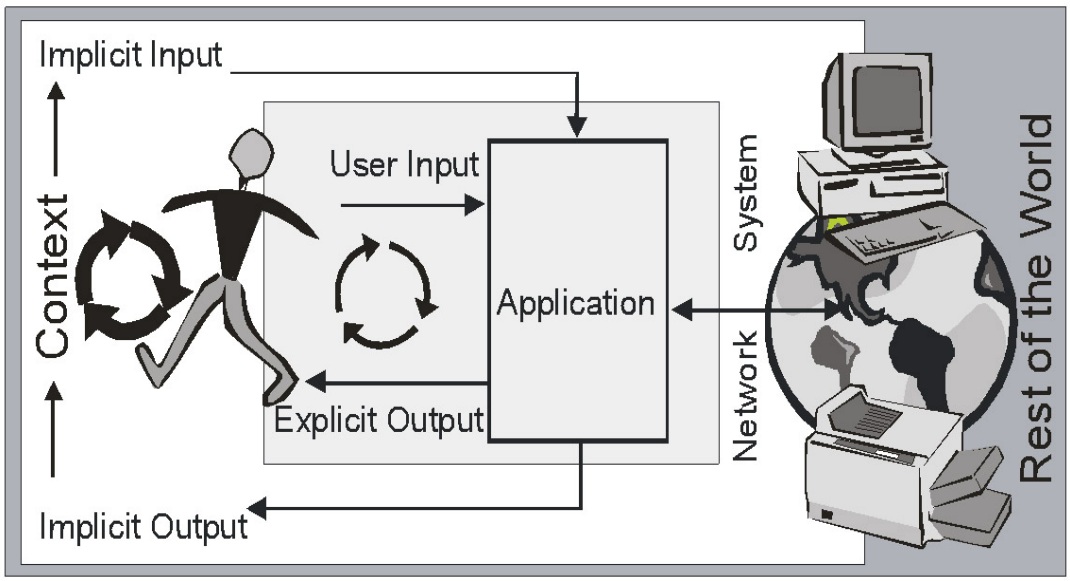

Figure 9.1 Implicit human computer interaction model

Applications that make use of iHCI take the context into account as implicit input and also have an influence on the environment by implicit output.

The proposed model is centered on the standard model in HCI where the user is engaged with an application by a recurrent process of input and output. In the iHCI model the user's centre of attention is the context – the physical environment where the task is performed. The interaction with the physical environment is also used to acquire implicit input. The environment of the user can be changed and influenced by the iHCI application.

The system and also the network are to some extent part of the context but are also accessible by the application directly.

9.4 Application Areas for Sensor-based Context-Awareness and iHCI

Implicit HCI is applicable in a great number of application areas and offers solutions in different problem domains. Especially for systems that should not distract the user from the main task and the interaction in the physical iHCI is of particular interest. As there are numerous specific domains and application areas the following subsection discusses these by considering classes of applications.

9.4.1 Proactive Applications, Trigger and Control

Using events or more general situations to trigger the start of applications is a common approach for using context and widely discussed and published [16], [17]. In most of these applications there is direct connection between the context and the application that is executed.

Starting and stopping represents the minimal proactive application. Further typical applications are warning systems and control systems that carry out a predefined action when certain context is recognized, e.g. thresholds are violated.

Selecting applications based on the current context is a further approach. A typical example is to have a device that is general purpose but becomes a specific information appliance depending on the context. One example is a PDA that runs its applications automatically according the context, e.g. when the PDA is close to a phone it runs the phone book application, in the supermarket the shopping list application is executed, and in the living room it becomes a remote control.

Using the current context information as parameter for proactive applications is a further approach. The behaviour of the application is then changed according to context. A simple example of this type of application is a navigation system. The context information – e.g. the current position and ground speed – is provided as parameter to the application.

The application uses this information to provide the appropriate information (e.g. a map centered to the current position using a scale appropriate for the travel speed). A further example is the use of context information to set default values so that they fit the current situation, e.g. in meeting minutes the form is already preset with appropriate default values for time, date, location, and participants. This type of application is closely related to applications that generate meta data.

A general and severe problem that occurs in this type of applications is the way how implicit and explicit user interaction goes together, see [18]. The basic question is how to resolve conflicting inputs? And furthermore how is it possible to achieve stability in the user interface without confusing the user. E.g. when a device is showing different behaviour depending on the situation and the user does not understand why the system behaves differently and in which way it might lead to confusion and frustration. It is therefore central to build user interfaces where the proactive behaviour of the system is understandable and predictable by the user even if the details are hidden (e.g. someone does not know how the automatic outdoor light works in detail but has a simple model of the reaction to expect when walking by during the night). In most cases it is also important to provide some way of allowing manual overwrite – where the user and not the context defines the parameters.

9.4.2 Adaptive UIs

Having information on the current situation available it becomes possible to build user interfaces that adapt to context. This is in particular interesting with regard to physical changes in the environment. Here again it is useful to draw a comparison with information appliances.

When designing a conventional information appliance the context of use is taken into account at design time. Assumptions about potential users and usage scenarios are made in the design process. Based on this analysis the user interface is created to support the anticipated use in an optimal way. Examples become obvious when comparing the design of mobile computing systems that are primarily targeted at different groups, e.g. PDAs for managers, Game-PDAs for kids, rugged mobile computers for harsh environments, and devices used for

fieldwork. These examples show that the context of use drives hardware design decisions (e.g. type of display, batteries, casing, and number of buttons) and software issues (e.g. visualization, menu structure, and use of colours).

In systems were context is available during runtime it becomes feasible to adjust the software part of the UI at runtime. In a very general view the requirements for the UI are dependent on the application, the UI hardware available, the user and the context. The requirements defined by the application may be quality parameters for the visualization of certain content. The UI can be a single device with specific properties or a distributed configurable UI system with various input and output options. The requirements defined by the situation, in particular context and user, may vary a lot. Examples are:

- That in the event of danger it is essential to provide information in a simple and quick to recognise way to the user
- When the user is engaged in a task it should not be necessary to move the focus in the real world in order to interact with the system. This can be archived by selecting the right display in a multi-display environment
- Privacy issues are a further concern. Interaction and visualization should be realised in a way to preserve the user's privacy depending on the situation.

A variety of challenges are evolving from the topic of adaptive UIs. The following two areas show exemplarily the problem domain.

UI adaptation for Distributed Settings

In environments where there is a choice of input and output devices it becomes central to find the right input and output devices for a specific application in a given situation. In an experiment where web content, such as text, images, audio-clips, and videos are distributed in a display rich environment we realized that context is a key concept for determining the appropriate configuration [19]. In particular to implement a system where the user is not surprised where the content will turn up is rather difficult.

UI adaptation in a Single Display

Adapting the details in a single user interface a runtime is a further big challenge. Here in particular adaptation of visual and acoustic properties according to a situation is a central issue. Simple examples that are by now available in different commercial products are the adjustment of the volume according to the environmental sound level and the regulation of backlight depending on the ambient light level. We carried out experiments where fonts and the font size in a visual interface became dependent on the situation. Mainly dependent on the user's activity the size of the font was changed. In a stationary setting the font was small whereas when the user was walking the font was made larger to enhance readability [20]. The orientation aware display described in [21] belongs also in this category.

9.4.3 User Interruption

Mobile computing devices and in particular communication device are designed to accompany the user and to notify the user about certain events. On a basic observation two types of

notification events can be discriminated. One type is pre-scheduled events, such as calendar entries that are specified to notify the user at a certain time. The other type is interruptions that are triggered by something else, e.g. a phone call from someone or a warning that batteries are low.

For both types it is interesting to exploit context for selecting the communication channel (e.g. visual, tactile, and acoustic) that is used to notify the user. Based on the context the intensity (e.g. volume, size of visual note) of the notification can be selected.

Especially in the area of wearable computing these issues are of major interest. In the project TEA several experiments have been carried out to assess how sensor based context can be used to modify notification interfaces at run-time [22].

In the case of pre-scheduled events context can be valuable to find the right time for delivery. In communication between humans rules for interruption are implicitly shared.

Depending on the urgency of the notification a suitable time can be found for delivery.

Context can enable devices to mimic this behaviour as reported in [23]. We showed similar findings in [24]. For pre-scheduled events context can also help to determine whether or not there is still need for this particular notification or if it is already void. A simple example is a context aware meeting reminder; when the user is already in the meeting (e.g. in a given room together with certain people) there is no need to remind her to go there.

9.4.4 Communication Application

Context information can help to enhance remote communication between people. Sharing of context information between people can avoid embarrassing situations for communication partners, as often the social environment determines what form of communication is acceptable (e.g. using a mobile phone during a church service is still embarrassing to the receiver of the call and most likely also to the caller). The acceptance for communication is also dependent on the current task a user is doing and in particular on the cognitive load.

In general there are two areas that can be discriminated:

- Context to filter communication. The basic idea is to filter communication dependent on the context. For each possible context filter properties are defined to determine the behaviour. This approach was taken in the TEA project. Similarly dependent on the current context the most relevant information for this particular situation is selected. Location-aware systems often centre on this concept [22]
- Context as communication mediator In many application domains automated filtering is rather difficult and often errors are not acceptable. In these cases where even a performance of 99% is not acceptable to the user context can become a mediator for communication partners. Setting up a phone conversation is due to a lack of context very different for a face to face communication. Phone calls “hit” the receiver often at an inconvenient point in time. Using – and especially sharing context – can help to ease the problem as we demonstrated in [22]. In this experiment the called party provides automatically some abstract context information to the caller. By this means the caller can decide whether or not it is appropriate to proceed with calling or not.

A number of issues that have to do with communication are close to user interruption as outlined earlier. In different cases they are also relevant for the management of resources as described in the next section.

9.4.5 Resource Management

Using resources dependent on the context and in particular on the location was a main motivation in the early attempts of using context [25]. An often used example is to automatically detect the printer that is close to the current whereabouts of the user.

Using resources that are physically close or in proximity of the user is central to this type of applications. The concept of physical proximity and the use of physical space as a criteria for ordering items and accessing them is a very natural concept for humans [26].

When taking into account the variety of context information that can be made available further application areas emerge. Especially in the domain of communication it is important to select resources that best meet the requirements of the current situation. E.g. dependent on the available battery power, the networks close by, and the requirements of the application the appropriate communication medium is chosen. Similarly the processing resources can be used context dependent.

This category of applications can be characterized as applications that use context to detect and find appropriate resource in a given situation as well as to adjust the use of resource to mach the requirements of the context.

9.4.6 Generation of Meta Data, Capture

Data hold in computer systems is often tagged with meta information – sometimes visible in the interface and also on system level. A typical example is the file system; the data contained in files is also associated with meta data such as the file name, the time and date of creation, and information on ownership and access control. Such meta data is either explicitly assigned by the user or taken out of the context of the system. The meta data is then available for the user (e.g. show files listed by creation data), for applications (e.g. UNIX make command) and also used by the system (e.g. granting access). Typically meta data is used as search and order criteria.

Using context that is outside the system further information becomes available and also usable as meta data. In Table 9.1 examples are given of how context can be used to retrieve documents.

Table 9.1 Using context meta data to retrieve documents

Context used	*Sample user query*
People around, Social context	Who was around when this document was created?
	Show all documents that were created while X was around.
Location information	Where was this document created?
	Show all documents that I have accessed while I was in London.
Location and time information	Show all documents that have been open together with this document. (same time and same location)
Environmental information	Show all documents that were created when it was cold.

Meta information can become an important part of the data stored. Applications that automatically capture context are central to the idea of Ubiquitous Computing [27] and also to

iHCI.

In the B2B domain (business to business e-commerce) we could show that long term capture of context information within business processes, such as transportation of goods and more general logistic, can enable new application scenarios [28].

This summary of application areas and the provision of examples shows that the iHCI model is widely applicable.

9.5 A Basic Problem: Pull vs. Push

As context offers additional information it becomes a major design decision how to incorporate the information in the system. The following discussion is related to issues of context based "information push and pull" as discussed in [29], but it also addresses the resulting implementation issues on system and application level.

9.5.1 Pulling for Context

In general "context pull" means that the consumer of context, e.g. the user, the application, or the system actively requests context that is required. In this mode the consumer controls when context is requested and used.

To implement applications that make use of context pulling context has the advantage that the application is in control when context is requested and when context has an influence on the system. The application programmer will build applications in a way that context is pulled at a convenient point in time for the application. Typically when a change in the interfaces appears anyway (e.g. due to an explicit interaction) context is requested and also taken into account. This can enhance and calm visual interfaces [30]. Another common option is that context is pulled when the application has time anyway and the received context is used later at an appropriate time.

The disadvantage of using a pull approach is that the information must be requested at least in the interval in which it could affect the application. Especially when implementing systems where the change of context is less frequent than the possible update interval in the application this is a waste of communication resource. When devices are using a wireless communication this can be a costly option. Having many consumers that request periodically context information from a supplier can also create a severe scalability problem.

9.5.2 Getting Context Pushed

In contrast "context push" describes a mode where the context producing entity provides context to possible consumers. The decision when to supply the context is up to the context provider in the system.

For the context provider this makes distributing context straightforward. Always when new context is available this is pushed to potential receivers. However this leaves the question open who are the potential receivers? Possible options to this issue are to deploy a subscriber model where a receiver can subscribe to a type of context [31] or to use broadcast as a general way of distributing context [32].

The push-model has the advantage for the context consumer does not actively have to query for context. However in terms of implementation this can be also an enormous

drawback. As context information can potentially arrive at any time this has to be taken into account. On a system level this requires a way of handling interrupts or multitasking functionality. This can especially be a problem for small microcontroller based systems with minimal system capabilities. The application has to store or buffer incoming context information to a point where they can be processed or presented. In cases where systems are connected wireless it requires that the receiver is available all the time because context information can be pushed at any time.

9.5.3 Combining Push and Pull

There are various options how to combine the approaches introduced above. An option is to introduce intermediate components such as proxies or subsystems that offer push or pull interfaces to consumers and producers depending on their requirements.

Consider the following example. An application can only make use of context at certain points in the program. For the application programmer it is the easiest option to pull at these points for context information, e.g. each time a transition in the interface is made.

This is possibly very often. However when assuming that the application runs on a device that is connected to the network wirelessly and that context changes happen rarely using context pull becomes on a system level questionable. Introducing a subsystem running on the mobile device that offers applications a pull interface but acts as a push receiver towards the network is a useful option. In this way programming model is kept simple and also the network traffic is reduced.

9.6 Humans and Invisible Computing

The notion of invisibility and disappearing computing is common to the vision of Ubiquitous Computing and ambient intelligence. Invisibility is not primarily a physical property of systems; often it is not even clearly related to the properties of a system. In this section the factors that influence the perception of invisibility are discussed. Investigating the effect of making everyday artefacts part of the digital world brings up the inherent dilemma - invisibility vs. added value.

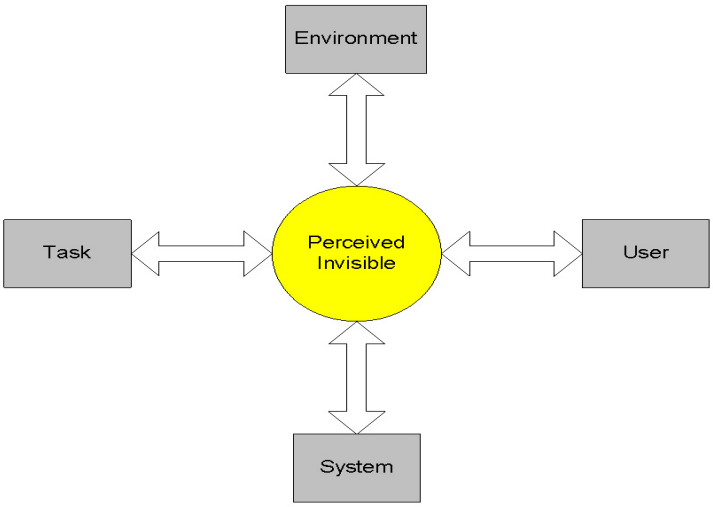

Figure 9.2 The factors that influence the perceived invisibility

9.6.1 How to Perceive Invisibility

It is not disputed that invisibility is a psychological phenomenon experienced when using a system while doing a task. It is about the human's perception of a particular system in a certain environment. Taking this into account invisibility has four factors that have a major influence: the human, the system, the task, and the environment, see Figure 9.2.

Only the relationship between all of them can determine the degree of invisibility that is experience. Again the degree of invisibility is hard to assess. Going along with the Normans argument [5,p.52] that the system becomes a natural extension to the task the following test can be helpful. The simple question "*what are you doing?*" can help to reveal the basic relationship between the tool, the user and the task. If to this question already the tool is mentioned the tool is central to the user's attention. If only the task is mentioned the tool has some degree of invisibility to the user. By detailing the question further: *How are you doing the task?* and *What steps are you performing to accomplish the task?* eventually the tool will be mentioned.

These questions can help to understand how much the tool is on the user's mind and how much she is taking the tool for granted and concentrating on the task. But in the same way the weakness of the concept of invisibility becomes obvious. Imagine you ask two people who are writing a text document. One person who is writing using the text based Unix programs *vi* and *latex*, the other one using a graphical word processor on a Apple Mac. Assuming that both have been using the system for a number of years the answers – and also their psychological perception of their tool – will in many cases not differ much.

Both will probably have formed a relationship with the tool so that it is used subconsciously.

This gives evidence that degree of invisibility perceived is strongly related to the familiarity of a tool for solving a particular task. This puts into perspective the notion of a "natural extension" [5,p.98] and the idea of "weave themselves into the fabrics of everyday life" [7] as this could be achieved by training the user. For many tasks there are no natural ways of doing it, take manual writing – children spend years in school to learn it.

Nevertheless in many cultures writing is considered to be natural.

Basically invisibility to some degree can be achieved for any tool – it doesn't matter how awkward it is – if the user spend enough time using it. This notion of invisibility does not relate to the basic ideas of Ubiquitous Computing. Therefore when considering systems the immediate invisibility is an interesting criterion. This is the question about how obviously can the tool be used to solve a task building on the common knowledge a user has.

9.6.2 The Invisibility Dilemma

The physical disappearance and in particular the integration has also an effect on the user's perception. Especially when digitally enhancing artefacts that are known and used in everyday live the physical invisibility of the technology plays a significant role.

When building computing and communication technology into everyday objects – and specifically technology for context-acquisition – there are two conflicting goals that pull the design in opposite direction:

- *Goal 1: invisible integration*. The technology that is needed to make everyday artefacts a part of the digital world should be invisible. The expression of the artefact should not be

changed by technology. With regard to the usage of the object there should be no change to the behaviour – the technology should be completely transparent.

- *Goal 2: added value.* When digitally enhancing everyday artefacts there should be an added value for the user. The added value can be on the artefacts themselves or in the overall system.

As we investigated in the project Mediacup [33] these goals appear in the first place not conflicting. In particular assuming the constellation that the artefact is enhanced and the added value is in the backend (e.g. coffee cup provides location of the user and on a map of the building activities are visualized). However the first goal also includes that people do not change their behaviour as the technology is transparent. But offering added value will stimulate human creativity to exploit what is available.

Even if an artefact only senses information and provides this to the system it becomes a handle for the user to manipulate the system. As humans are creative to find ways to use technology in a way to efficiently achieve their goals, they will change their behaviour to optimally exploit the capabilities of the system.

This does not question the design of transparent and invisible system but the designers should be aware that people will make use of the added value provided – often even in an unintended way. Examples are objects that become location tokens for their users (ActiveBadge [34], MediaCup [35]) and they will be used as such – and not necessarily in a similar way as their non digital counterparts (a badge and a mug).

9.7 Discussion

In some models and implementations context is seen as just another form of user interface component that provides information to the system. In such approaches context information is treated similarly to events that originate in the graphical user interface. This approach however has drawbacks concerning the predictability of the user interface. When a user interacts explicitly with a user interface (e.g. by pressing a button) she expects something to happen (e.g. get a different form onto the screen). The user action relates directly to the change interface and is therefore easily understandable. When context is used similarly than a change happens without an explicit interaction beforehand – in this case the user may be surprised by the change in the system. When designing such interfaces it is important to be aware of the difference between an event explicitly generated by the user and information acquired from context. In case of explicit interaction user's expectations are clearly related to the interaction carried out (e.g. pressing the back button has a semantic assigned). In the case of a context driven change the semantic of the situation or context (e.g. lowering oneself in the armchair in front of the TV) is often ambiguous. In [15] further issues about the integration and representation of ambiguity in user interfaces are elaborated. Therefore it is important to distinguish these concepts when designing an interactive system and to provide hints in the interface so that the user has a chance to find out what action provoked the reaction of the system.

One very basic question to address when designing interactive context aware systems is the trade-off between stability in the interface and adaptation of the interface. The main argument for stability is that humans picture the interface and know where to look for a function. This spatial memorizing becomes much harder or even impossible if interface keeps changing. The

counter argument is that if adaptation works well the right functions are always at hand and there is no need to memorize where they are. Depending on the system that is designed one or the other argument is more important. For the design of context aware systems these issues should be taken into account and the trade-off should be assessed.

9.8 Summary and Conclusion

In Ubiquitous Computing and ambient intelligence most systems consider that there are humans in the loop. These systems are obviously interactive. As humans interact in many ways with their environment the term interactive application goes beyond the well established user interfaces. Traditional user interaction is in most cases dialog oriented whereas the communication and interaction between humans and humans and also between humans and their environment is much richer. In particular the situation in which a communication takes place has a significant role for the common understanding.

Taking the environment into account a new interaction model can be established – regarding explicit as well as implicit interaction. This model can be used to explain different application areas of context aware systems.

Implementing systems that make use of context information require basic design decisions on the way context is integrated. Basically pushing context to the system and eventually to the UI and pulling context from the resource are the two pure options. In most real system the design will result in a combination of both. Important factors on push and pull are the system architecture and distribution as well as the constraints on the user interface.

Invisibility is not a property of the system it is rather a complex relation between the user, the system, the environment, and the task carried out. The idea of invisibility is dependent on the knowledge of the potential user and her expectations on a “natural tool”.

Integrating computing technology into everyday objects also addresses the issue of physical disappearance. But building invisible systems the designer is always subject to the dilemma between true invisibility and added value. Including technology that provides added value of a certain form will in many cases trigger the ingenuity of the user and make her use the object differently. Object and artefacts which could be used for their original purpose transparently become different objects because they are a manipulator for the digital world.

References

[1] A. Dix, J. Finlay, G. Abowd, and R. Beale, *Human Computer Interaction*, 2a Ed., Prentice Hall Europe, 1998.

[2] R. B. Miller, Response time in man-computer conversational transactions, *Proc. AFIPS Fall Joint Computer Conference* Vol. **33** (1968), 267-27.

[3] J. Nielsen, Usability Engineering, Morgan Kaufmann, San Francisco, 1994. Excerpt on *Response Times: The Three Important Limits*. Online: http://www.useit.com/papers/responsetime.html.

[4] E. Bergman, (Ed.), Information Appliances and Beyond, Morgan Kaufmann Publishers, February, 2000.

[5] A. D. Norman, The Invisible Computer, Cambridge, Massachusetts; MIT Press. 1998.

[6] B. Shneiderman, Direct manipulation: A step beyond programming languages, *IEEE Computer*, **16**(8), (1983) 57-69.

[7] M. Weiser, The Computer for the 21st Century, *Scientific American*, **265**(3) (1991) 94-104.

[8] M. Weiser & J. S. Brown, The coming age of calm technology, in P. J. Denning & R. M. Metcalfe (Eds.), *Beyond calculation: The next fifty years of computing*, New York, NY, 1998, 75-85. Online:

http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm.
[9] J. Wejchert, The Disappearing Computer, Information Document, IST Call for proposals, European Commission, Future and Emerging Technologies, February 2000. Online: http://www.disappearing-computer.net/mission.htmlS. Wright, Evolution in Mendelian Populations, Genetics, 1931, 16-97.
[10] D. Lenat, The Dimensions of Context Space, Technical report, CYCorp, October 1998. Invited talk at the conference Context 99. Online: http://www.cyc.com/context-space.rtf.
[11] H. W. Gellersen, A. Schmidt, M. Beigl, Ambient Media for Peripheral Information Display, *Personal Technologies* **3**(4) (1999) 199-208.
[12] A. Schmidt, H. W. Gellersen, Visitor Awareness in the Web, *10th World-Wide Web Conference* (WWW10), W3C, Hongkong, May 2001.
[13] H. W. Gellersen, A. Schmidt, Look who's visiting: supporting visitor awareness in the web, *International Journal of Human Computer Studies* (January 2002) IJHCS **56**(1), 25-46.
[14] A. Dix, Beyond intention -pushing boundaries with incidental interaction, Proceedings of Building Bridges: Interdisciplinary Context-Sensitive Computing, Glasgow University, 9 Sept 2002.
[15] J. Mankoff, An architecture and interaction techniques for handling ambiguity in recognition-based input, Ph. D. Thesis Dissertation, College of Computing, Georgia Tech, May 2001.
[16] W. N. Schilit, N. I. Adams, and R. Want, Context-aware Computing Applications, in the Proceedings of the *1st International Workshop on Mobile Computing Systems and Applications*, 85-90, Santa Cruz, CA, IEEE. December 8-9, 1994.
[17] P. J. Brown, J. D. Bovey, and X. Chen, Context-Aware Applications: From the Laboratory to the Marketplace, *IEEE Personal Communications*, **4**(5), (1997) 58-64.
[18] K. Cheverst, N. Davies, K. Mitchell, and C. Efstratiou, Using Context as a Crystal Ball: Rewards and Pitfalls, *Personal Technologies Journal*, **3** (5), (2001) 8-11.
[19] M. Beigl, A. Schmidt, M. Lauff, H. W. Gellersen, The UbicompBrowser, *4th ERCIM Workshop on User Interfaces for All, Sweden*, 19-21 October 1998.
[20] A. Schmidt, Implicit Human Computer Interaction Through Context, *Personal Technologies Volume* **4**(2&3) (2000) 191-199.
[21] A. Schmidt, M. Beigl, and H. W. Gellersen, There is More to Context than Location, in *Interactive Applications of Mobile Computing, Rostock*, (1998) Germany, 24-25.
[22] A. Schmidt, A. Takaluoma, and J. Mntyjrvi, Context-Aware Telephony *over WAP*, Springer-Verlag, London, Ltd., Personal Technologies, Vol 4, 225-229. Short paper presented at Handheld and Ubiquitous Computing (HUC2k), HP Labs, Bristol, UK, September 2000.
[23] N. Sawhney, C. Schmandt, Speaking and Listening on the Run: Design for Wearable Audio Computing, Proceedings of the International Symposium on Wearable Computing, Pittsburgh, Pennsylvania, 19-20, October 1998.
[24] A. Schmidt, H. W. Gellersen, and M. Beigl, A Wearable Context-Awareness Component - Finally a Good Reason to Wear a Tie, *IEEE Proceedings of the third International Symposium on Wearable Computers*, San Francisco, 18-19 Oct. 1999, 176-177.
[25] W. N. Schilit, A System for Context-Aware Mobile Computing, Ph.D. Thesis, Columbia University, New York, 1995.
[26] D. Kirsh, *The* Intelligent Use of Space, *Journal of Artificial Intelligence*, **73** (1-2), (1995) 31-68. Online: http://icl-server.ucsd.edu/~kirsh/Articles/Space/AIJ1.html.
[27] G. D. Abowd, Classroom 2000: An Experiment with the Instrumentation of a Living Educational Environment, *IBM Systems Journal, Special issue on Pervasive Computing*, **38** (4), (1999) 508-530.
[28] A. Thede, A. Schmidt, C. Merz, Integration of goods delivery supervision into E-Commerce supply chain, *Second International Workshop on Electronic Commerce* (WELCOM'01), Heidelberg, Germany, November 16-17, 2001.
[29] K. Cheverst, N. Davies, K. Mitchell and P. Smith, Exploring Context-Aware Information Push, in *Proceedings of Third International Workshop on Human Computer Interaction with Mobile Devices* (Mobile HCI), 10 Sept 2001, At IHM-HCI 2001, Lille, France. 2001.
[30] S. S. Intille, Change Blind Information Display for Ubiquitous Computing Environments, UBICOMP 2002, *International Conference on Ubiquitous Computing*, Goteborg, Sweden, September 2002.
[31] D. Salber, A. K. Dey, and G. D. Abowd, The Context Toolkit: Aiding the Development of Context-Enabled Applications, *Proceedings of the Conference on Human Factors in Computing Systems* (CHI '99), Pittsburgh, PA, May 15-20, 1999, 434-441.
[32] The Smart-Its Project, European Disappearing Computer Project, 2002. online: http://www.smart-its.org/
[33] H-W. Gellersen, A. Schmidt, M. Beigl, Multi-Sensor Context-Awareness in Mobile Devices and Smart

Artefacts, *ACM journal Mobile Networks and Applications* (MONET), **7**(5) (2002).

[34] R. Want, A. Hopper, V. Falcao, and J. Gibbons, The Active Badge Location System, *ACM Transcation on Information Systems* **10** (1), (1992) 42-47.

[35] M. Beigl, H-W. Gellersen, A. Schmidt, MediaCups: Experience with Design and Use of Computer-Augmented Everyday Objects, *Computer Networks*, Special Issue on Pervasive Computing, Elsevier, **35** (4) (March 2001) Elsevier, 401-409.