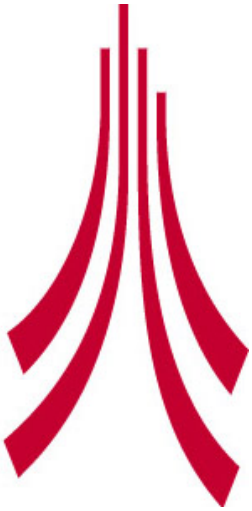
Ubiquitous Computing – Computing in Context



A thesis submitted to Lancaster University

for the degree of

Ph.D. in Computer Science,

November, 2002

Albrecht Schmidt, MSc

Computing Department,

Lancaster University, U.K.

Abstract

Ubiquitous Computing – Computing in Context

Albrecht Schmidt, MSc

Computing Department,
Lancaster University,
England, U.K.

Submitted for the degree of Doctor of Philosophy,
November, 2002

Computers have advanced beyond the desktop into many parts of everyday life. Ubiquitous Computing is inevitably computing in context: it takes place in situations in the real world. So far most research, especially in mobile computing, has focused on enabling transparent use of computers, independently of the environment. An orthogonal research effort is to exploit context. The research reported here is investigating: how context can be acquired, distributed, and used and how it changes human computer interaction in Ubiquitous Computing.

Possible sensing technologies, in particular low level physical sensors, and perception techniques are assessed and their value for providing context in Ubiquitous Computing systems is analysed. Abstractions on sensor level, cue level, and context level are introduced, resulting in a flexible context acquisition architecture.

A bottom-up approach for modelling context aware systems is introduced. This makes use of the fact that context or domain knowledge is more general on the level of artefacts, than on the system level. The creation of context aware systems, based on this approach, is then investigated using the method of prototyping. To generalise and communicate results, a pattern language for context aware systems is suggested.

As context acquisition systems are mostly specific to a certain task, building such systems involves designing and building hardware and software. The research presented here shows methods, architectures, and tools to make the development process more efficient. The Smart-Its platform, a rapid prototyping system for context-aware Ubiquitous Computing systems, is introduced and use experience is reported.

The observation that context naturally surrounds us, led to the development of a communication platform. This platform provides an effective means to distribute and receive information based on spatial and temporal relationships of components.

In this research the notion of implicit human computer interaction, and in particular the use of context information as implicit input, is introduced. The implications on the user interface and on the human computer interaction process are analysed, as context is fundamentally different from events in user interfaces.

Finally the research presents an overview on how Ubiquitous Computing systems can be evaluated. Different techniques are assessed, and the concept of probing users and developers with prototypes is presented.

Declaration

The work reported in this thesis has not been submitted in support of an application for another degree at this or any another university.

Excerpts of this thesis have been published in journals, conference and workshop articles as well as research deliverables and patents, most notably [Beigl,98], [Beigl,01], [Gellersen,99a], [Gellersen,00], [Gellersen,02], [Gellersen,02a], [Laerhoven,02], [Patent,01], [Patent,02], [Schmidt,98], [Schmidt,99], [Schmidt,99a], [Schmidt,99b], [Schmidt,99c], [Schmidt,00], [Schmidt,00a], [Schmidt,00b], [Schmidt,01], [Schmidt,01a], [Schmidt,02], [Schmidt,02a], and [Thede,01].

_____

Lancaster, November 2002, Albrecht Schmidt

Acknowledgements

Hans-Werner Gellersen has given me the unique opportunity to work with him in two exciting places to do research in Ubicomp: at TecO, at the University of Karlsruhe in Germany, and at Lancaster University in the UK. He advised, encouraged, and inspired my research and most importantly became a close friend. I feel extremely lucky to have worked under his guidance and I am thankful for all the confidence he had in me, and for freedom and responsibilities I received in my work.

I owe a great debt to Prof. Krüger who supervised my work during the time in Karlsruhe. He gave me the opportunity to lecture, and taught me a lot about research, teaching, and life. I appreciate his unselfish generosity of letting me go to Lancaster and also that he did everything possible to make the move easy.

I would like to thank all my colleagues at TecO. Many ideas that resulted in the research presented in this thesis had their origins in (late night) discussions with Michael Beigl. At TecO I have been fortunate to always have students around with a genuine interest in Ubiquitous Computing research. In particular I appreciate Christian Decker's never ending string of ideas and Martin Strohbach's insightful critical remarks.

I appreciate very much the way I have been welcomed by my new colleagues at Lancaster University and for all the support I have received. I like to thank Paul Rayson, who I have shared office with and from whom I have picked up some knowledge on Corpus Linguistic, for his patience with all my 'foreigner's questions'. I also like to extend my thanks to my colleague Kristof van Laerhoven and the

Ubiquitous Computing group. Special thanks are to Mark Taylor who volunteered to proof-read the thesis.

Most of the research was carried out in European projects and in cooperation with industrial research laboratories. I have been lucky to meet the most interesting people through this path, but there are too many to enumerate. Many of them had an important impact on the way I think about research, to name a few: Walter van Velde (Starlab), Oliver Frick (SAP), Antti Takaluoma (Nokia), and Bernd Schiele (ETH Zurich). A sincere thank you to all.

I would like to thank my parents for their love, encouragement, and support throughout my education, which was, especially in the very beginning, not always easy. I am also thankful for all the support we received from my parents-in-law.

I would like to express my deepest gratitude to my dear wife Petra Dollinger, who made many compromises to let me finish my Ph.D. in the UK. Without your friendship and love this thesis would not have been completed. Finally, I like to thank my dear daughter Vivien Georgie for the encouragement she has given to me over the last year, probably without knowing it.

Thanks to all of you!

A lot of the research presented in this thesis was funded by:

- The European Union Information Technologies Programme, Esprit Project 26900.
- Disappearing Computer Initiative, Information Society Technologies, European Union.
- EQUATOR. Interdisciplinary Research Collaboration (IRC) supported by EPSRC.
- SAP Corporate Research, Karlsruhe, Germany.

Dedication

To my parents, for teaching me what is important in life.

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Overview

Research in Ubiquitous Computing has arrived at a crossroad: A point of convergence where a technology proliferated environment meets with the ability of people to interact with, and make use of, the possibilities that this technology creates. Advances in the various fields of technology allow us to create artefacts and environments that provide computing and communication resources. The understanding of how humans will interact and make use of such systems is however largely unresolved and often not addressed in current research. A key to understanding such systems and their use is the observation that humans implicitly interact in context with their environments including technology.

The task of making this context information available to components in computer systems has become a prerequisite to advancing human-computer interaction processes in Ubiquitous Computing. Context awareness, or more specifically how to create applications that are context aware, is a central issue to Ubiquitous Computing research. Such research raises questions on context acquisition, context representation, distribution and abstraction, as well as programming paradigms, development support, and implications on human-computer interaction in general.

The research presented in this thesis concentrates on some of these issues. First the question of how to acquire context in a Ubiquitous Computing environment using

simple sensors is addressed. Then a bottom-up approach for modelling context-aware artefacts is introduced. Prototyping various context sensing devices and generalising to patterns demonstrates the feasibility and applicability of the approach. Assessment is further done by evaluating the ways in which providing sensor based context can be made easier by using methods, libraries, tools, and physical building blocks. Given that context is made available, the issue of distribution is addressed. From this, a new human computer interaction model that includes context is proposed; a model that also takes account of explicit and implicit user interaction. Finally ways are addressed in which Ubiquitous Computing systems can be evaluated.

The research is motivated by the idea of a human centred approach to Ubiquitous Computing, as outlined in the following paragraphs. The research reported however includes hardware, software, and communication issues, as well as topics related to interaction.

1.2 Ubiquitous Computing

Approaches in Computer Science in the last 50 years can be related to the quantitative relationship between computers and humans. At the very beginning many people shared a single computer, then the idea that each user has a single computer significantly changed the way people used computer systems. In the last decade this changed further into a many-to-one relationship, where one user has many computers, or at least devices with processing capabilities available to, and surrounding a single user. This recently started era is referred to as Ubiquitous Computing; however, Ubiquitous Computing raises many issues beyond the quantitative relationship between computer and user [Weiser,91], [Weiser,96].

From a Human computer interaction (HCI) viewpoint Ubiquitous Computing describes the phenomenon of interacting in context with artefacts and environments that are interwoven with processing and communication capabilities. Here the focus is to empower humans interacting in such an environment and to enhance their interactive experience. This viewpoint is in the tradition of early interactive computer systems; as an early innovator in this area Douglas Engelbart, described his work as augmenting human intellect contrasting with work that focused on automating tasks [Engelbart,62]. Over the last few years there have been observations of a further shift

towards creating a good experience for the user [Newman,01]. From a technical perspective many computer systems have already achieved efficiency. However, creating a good experience for a user can improve their perception of their work and so ultimately make the process of using a computer system more effective and pleasurable. This is especially true of tasks where creativity of the user is essential (e.g. writing, designing, construction) or where the task itself is recreational (e.g. games): In these examples experience becomes the main factor. Using conventional interfaces, such as desktop computers with screen, keyboard, mouse, and speakers limits the design space for creating experience; however, including the real environment as part of interaction for a computer system offers many interesting possibilities.

A major challenge in Ubiquitous Computing is physical integration and embedding of computing and communication technology into environments and artefacts. Such developments lead to 'augmented artefacts', raising issues beyond the physical integration. Embedding technology into everyday artefacts also inevitably implies embedding the "computer" into tasks done by the user. This leads to new research challenges and further questions.

- What is the consequence of artefacts and environments becoming an integral part of the "computer"?
- How is it possible and even pleasant to interact with a system where many artefacts and the environment is a part of the "human computer interface"?
- Where is the application and how do we influence and interact with an application when each part of the "computer" and of "human computer interface" is a potentially a part of many applications?

These issues that are central to research in Ubiquitous Computing lead to the research addressed in this thesis. Context, especially making context available to the "computer", is regarded as a dominant prerequisite to advance on these questions [Abowd,00].

In an optimal setting the technology disappears so that the "computer" and the "human computer interface" are hidden at least in the perception of the human. This implies that the user is primarily doing a task and is not aware of operating a computer system.

1.3 Interaction in Ubiquitous Computing

The terms *calm computing* [Weiser,98], *invisible computing* [Norman,98], and *disappearing computer* [Wejchert,00] describe the user interface perspective on Ubiquitous Computing. As the interaction is interwoven with the user's actions the concept goes beyond the traditional understanding of a human computer interface towards describing the relationship between the user and their augmented environment.

Making the computer invisible is not a matter of size or a challenge of seamless integration of hardware, it's about how the human perceives the computer. To make the computer disappear (at least in the user's perception), the interaction has to be seamlessly integrated with the primary task of the user. The user still interacts with the tools that help them to do a certain job, but their focus is on the task itself. This is in contrast to typical usage of a computer as a tool, where the focus usually ends on the computer not on the task [Weiser,98]. In a Ubiquitous Computing paradigm tools are enhanced with processing and communication capabilities to help with achieving the task not drawing focus away from it.

Embedding interaction into tasks seems to be the obvious approach to take. However, when it comes to modelling and implementing this vision many unresolved issues appear. Using explicit interaction, as in conversational computer systems, there is provision for a choice of varying modalities. The interaction designer can chose from command line user interfaces, graphical user interfaces, and speech and gesture interfaces. Independent of the modality the user still is required to interact with a computer. Another issue that makes conversional interaction methods difficult is that interface components can be physically distributed and dependent on each other. On the other hand as there is also the potential for many applications to run at the same time, inputs have to be directed to a particular one. Using solely this approach would

inevitably result in a complex interface and require a great deal of the user's attention, which is regarded as one of the most precious resources because it is limited.

When interaction is embedded it happens in context. The physical environment, the situation, the role of the user, their relation to other users and to the environment, and their goals and preferences can all be rich source of information. Using this information when making a system context-aware can make the explicit interaction process much easier or even eliminate the need for explicit interaction. A reduction in explicit interaction will also reduce the demands on the user's attention, assuming that the system gets it right. This raises a further issue: how to acquire and provide context?

1.4 Context-Awareness is an Enabling Technology

In Ubiquitous Computing, interaction with computers is inevitably in context and in most cases context matters for not only the users directly, but it also matters indirectly for the system. The user's expectations about a system and their anticipation of the reaction of a system that they are interacting with, is highly dependent on the situation and environment, as well as on prior experience.

Interaction in the physical world is experienced from a very early age and the knowledge about the reaction of the environment accumulated over a lifetime. This knowledge allows intelligent behaviour; in particular the ability to predict the reaction that a certain action will provoke is a major advantage, and at large essential for survival. Many expectations are just extrapolated from previous experience, e.g. when operating a light switch on a bedside lamp in a hotel we expect that this particular light will switch on. We would be rather surprised if instead of the bedside light coming on the fan in the shower started or the radio in our car outside the hotel starts playing. Our expectations are based on experience and are essential to the way we live.

In Ubiquitous Computing environments, where the real world becomes a part of the computer and of the user interface, users expectations towards the system are also widely based on the experience of interaction in the real world. The designer however has a great freedom of how to design interaction processes in such systems. Many limitations inherent in conventional engineering are no longer an issue in Ubiquitous

Computing, in fact a networked switch could operate anything else that is networked. To make a system useful and give the user the feeling of being in charge of the system a switch should operate what the user anticipates in a particular situation.

This simple example of a switch shows that context is essential for building usable Ubiquitous Computing systems that respond in a way that is anticipated by the user. Context-awareness becomes a fundamental enabling technology for Ubiquitous Computing and is a key issue when creating computers that are invisible and disappear in terms of the user's perception. In these terms context-awareness goes beyond providing context information, it also requires understanding context and ultimately understanding situations.

1.5 Challenges in Context-Aware Computing

Examples, demonstrators, and prototypes have been used to demonstrate that context-awareness can enhance applications and systems. Typically location is sensed and then based on the location further assumptions about the more general context are made. As the concept of position and location is well understood, it also provides a powerful and easy to apply model for context-aware applications. In many cases however awareness based solely on location lacks information that can be of interest to a system for making it context-aware. If information beyond location information is required, further complexity is introduced.

The following issues are central research challenges in context-awareness:

- **Understanding the concept of context.**
  What does context mean and how is it connected to situations in the real world? There is still a fundamental lack of understanding in terms *how contexts relate to situations* and how general context information can be used to help enhance applications. This is also associated with the question of how to represent context in a universal way.

- **How to make use of context?**
  Assuming that context is available in a system the question *what is context useful for* becomes imminent: especially if contexts beyond location and available resources are considered. In this instance a central question is what

type of applications can be enhanced? When considering context as additional input, issues of reliability and ambiguity are important. Furthermore the relation between context and other inputs into the system and how they influence each other, have to be addressed. Ultimately this requires the smartness of the system to *understand* the context it is dealing with.

- **How to acquire context information?**
  Acquiring context is a prerequisite for any context-aware system. Generally context acquisition can be seen as the process where the real situation in the world is captured, the significant features are assessed, and an abstract representation is created, which is then provided to components in the system for further use. Approaches to acquire context are manifold and include computer vision, location tracking, sensor systems, and also more predictive approaches such as modelling users and their behaviour.

- **Connecting context acquisition to context use.**
  In a location-aware system there is a close relationship between context acquisition and context use, most often the location sensor is attached to the device using position as context. In this case the context representation is also agreed between these components. In more general environments context use and context acquisition is distributed. It can be assumed that context is provided for various applications, potentially in dynamic configurations. This makes it obvious that mechanisms to connect context acquisition and context use become essential. Here the challenge is twofold: *overcoming the distribution issue* by networking components and *agreeing on representations* that are useful for a multitude of components.

- **Understanding the influence on human computer interaction.**
  When systems are context-aware their behaviour is dependent on the context of use or the general situation of use. The ultimate goal is to make systems in such a way that they react as anticipated by the user. In real life however this creates complex problems, in particular if the system reacts differently from the users expectations. Two critical issues are *how can the user understand the system and its behaviour?* and *how to give the user control over the system?*

- **Support for building context-aware Ubiquitous Computing systems.**
  Context-awareness is an enabling technology for Ubiquitous Computing systems and therefore commonly required when realising such systems. To build Ubiquitous Computing environments efficiently, it is inevitable that we need to provide support for building context-aware applications. Up to now, there are many cases where the wheel is re-invented; where all the problems have to be solved over and over again in each system. Providing *support for context acquisition, context provision, and context* use will make the process of implementing context-aware applications much simpler.

- **Evaluation of context-aware system.**
  As context-aware systems are used in context, evaluation itself is also required *to be done in context*. In cases where functionality is only available and useful in a certain context it is required to create or simulate a particular situation that results in the wanted context in order to assess the system. Inducing a particular situation and context however may have also a significant effect on measures in the evaluation.

Many of these research issues are highly interconnected. Nevertheless some of the issues can be tackled fairly independently of some others. In the approach pursued in the course of research underlying this thesis, context-awareness is approached from a bottom-up perspective. In the bottom-up approach context acquisition and context use is related to artefacts. This approach cuts across several of the challenges above.

1.6 Awareness of Artefacts

Context and context-awareness is often regarded in a rather general way and hence the models and concepts that can be provided are on a very abstract level. When considering context on an abstract level, resulting models are generally applicable, but lack help for solving specific problems. In particular the question of what entity does the context relate to? (e.g. temperature of a room, motion of a handheld device, and load on a surface) becomes very important for context beyond location.

In this thesis the approach is taken to investigate contexts of artefacts. In this approach contexts are identified by regarding generic and intrinsic properties of artefacts. Using

this method of modelling context bottom-up provides a means to avoid the complexity faced when modelling context from a top-down perspective.

At a first glance it appears feasible to identify important intrinsic properties of an artefact with little effort. However when investing more time it seems to be an endless list. Therefore the quest for a generic set of properties is essential, but not straight forward. The question remains: which intrinsic properties are generic for an artefact? This problem is similar to modelling a digital representation of an artefact, as known from Object Oriented methods.

The proposed approach models, designs, and implements context-aware systems, by using a set of intrinsic and generic properties of artefacts that are part of the Ubiquitous Computing environment. This approach raises issues of distribution and fusion of context in such systems.

A successful implementation will result in artefacts that have a digital awareness. Here it is of particular interest that the digital self-awareness may be of no interest to the artefact itself (e.g. why does a coffee cup have to know its temperature if it can't do anything to keep the coffee warm?) but may be of great interest to other artefacts, devices, and applications in that environment.

1.7 Scope, Aims and Method

The objective of the research presented is to assess ways in which context-awareness can be facilitated in Ubiquitous Computing systems. In particular the focus is on using low-end, low-price computing and communication technology in order to identify solutions that could be economically deployed in everyday artefacts and environments in the near future.

The prime interest is on context acquisition using a variety of sensors. The aim is to provide an overview of possible sensing technologies and abstraction methods. A further goal is to find models, architectures, and methods that help to understand the field and ease the development process.

The methods used include systematically surveying literature and available information, designing and implementing prototypes to prove the feasibility of the

proposed ideas, creating models and concepts that generalise what was learned from the prototypes, and evaluation of the proposed solutions.

1.8 Novel Issues and Contribution

The thesis raises several novel issues in the field of Ubiquitous Computing. The main area of work is on context acquisition, context-awareness and human computer interaction. The major contributions are:

- An overview of sensors and sensing technology with respect of their applicability for Ubiquitous Computing environments, based on a prototypical assessment of various sensing technologies.

- The concept of bottom-up context-awareness, where generic intrinsic properties of artefacts are taken as the starting point to model context. Beyond the single artefact patterns for context-awareness have been identified.

- A framework, including methods, architectures, platforms, and libraries, that supports the design, simulation, implementation and maintenance of distributed context acquisition systems.

- The concept of implicit human computer interaction, as a prerequisite to invisible computing and disappearing user interfaces.

- An overview of evaluation techniques for Ubiquitous Computing systems.

1.9 Research Context

The research reported in this thesis is rooted in a number of different research projects, most importantly the following:

The European project ‘Technology for Enabling Awareness’ (TEA) was concerned with the development of a component that can provide context information beyond location. The objective was to assess what sensors, sensing systems, and algorithms can be used to facilitate this. As potential devices that use context mobile phones, PDAs, and wearable computers were anticipated. Further information and a list of publications can be found at [TEA,98].

In the European project ‘Smart-Its’ the focus is to provide a means for post-hoc augmenting artefacts with context acquisition technology. Specific areas of interest in the project are collective awareness and tools for rapid prototyping of Ubiquitous Computing Systems. More information is available at [SMART,02].

The interdisciplinary research collaboration ‘Equator’ focuses on the integration of physical and digital interaction. In particular in the project ‘Domestic Environments’ research is considering technologies that are compatible with everyday life in real world environments. For further information see [EQUATOR,02].

1.10 Thesis Outline

The thesis is structured in the following way. In chapter 2 the terms of Ubiquitous Computing, invisible computing, and disappearing computing are assessed. Then the concept of context and context-awareness is introduced and an overview of related work is presented.

Chapter 3 follows the basic question of how to acquire context information using sensors. In particular the relationship between situation, context and sensor data is examined. An overview of available sensing technologies, perception methods and algorithms is presented, and assessed for their suitability in Ubiquitous Computing environments. Based on the analyses a flexible and layered architecture for context acquisition is introduced.

The concepts of prototyping context aware artefacts and bottom-up context modelling are established in chapter 4. The basic approach of tying context to an artefact is introduced and the consequences are investigated. A number of prototypical implementations, following the approach, are presented. Based on these examples patterns of context-aware artefacts are identified.

Chapter 5 concentrates on issues relevant for supporting the design and implementation of context acquisition systems. Libraries and templates for the design of hardware, communication, and software are provided. A new method to ease design and implementation is introduced. Finally Smart-Its as a rapid prototyping platform for context acquisition is presented.

In chapter 6 one particular issue of context in Ubiquitous Computing environments is assessed: distribution. First the distributed nature of context and context acquisition is analysed. Then a distribution model and platform is introduced which has natural distribution properties built into the architecture.

The user interface perspective on Ubiquitous Computing and in particular on the usage of context is analysed in chapter 7. In particular the implications of invisible computing and disappearing computers on the human computer interface are assessed. Inspired by natural interaction the concept of implicit human computer interaction and its applications are introduced.

Chapter 8 presents the overall evaluation of the thesis. The chapter starts out with a quest for suitable evaluation methods assessing shortcomings of standard evaluation methods known from human computer interaction, mobile computing, software engineering, and AI in a Ubiquitous Computing environment. Evaluation methods related to prototyping are introduced and the hypotheses stated in the thesis are revisited and evaluated.

The conclusion in chapter 9 summarises the contributions made in the thesis, but also critically assesses the shortcomings and limitations detected in the course of the research. Furthermore new issues that have been surfacing while working on the thesis are addressed in the future work section of the chapter.

Chapter 2

Background and Related Work

Research in Ubiquitous Computing is very diverse as the field itself has not yet been clearly defined. Researchers from different communities that make efforts to understand and improve concepts, technologies, interaction and applications for computing beyond the desktop personal computer, undertake research in Ubiquitous Computing[1]. The research originates from many different areas such as mobile computing, distributed systems, human computer interaction, AI, design, embedded systems, processor design and computer architecture, material science, civil engineering and architecture. This very broad view on Ubiquitous Computing research is however not commonly shared.

A more widely accepted perception of Ubiquitous Computing research is that it involves an interdisciplinary element [Gellersen,99], [Thomas,00], [Abowd,01], [Borriello,02]. Developing and deploying technologies with a human centric view is a further characteristic aspect [Davies,02], [MIT,02].

Often research is carried out under a theme, in programs and initiatives providing a framework under which many different issues are investigated. Specific issues are explored in context of larger theme which may be a major research challenge in a

[1] This impression was backed up by looking at the overall submissions received by the HUC and Ubicomp conferences over the last four years. Information about the conferences can be found at http://www.ubicomp.org.

specific research field (e.g. mobile computing). Results that contribute to the specific field (e.g. a new ad-hoc protocol) also then contribute to Ubiquitous Computing (e.g. implications on the overall system and on the user's relationship with the system).

As themes, programs, and initiatives are central to research in ubiquitous computing so far, this chapter will present background and related work along these lines. First the visions for computing beyond the desktop PC are presented. Then more specifically the notion of context and how it evolves is discussed. Then an overview of research programs carried out at different institutions is given to provide an outline of the state of the art in context-aware computing. As the thesis is concerned in a large part with context acquisition, the means by which context is acquired are investigated and presented. Finally the issue of methodology and evaluation is addressed showing newly evolved approaches.

2.1 Visions: Computing Beyond the Desktop

Most people will still associate the term *computer* with a *Personal Computer* (PC) and its typical desktop use. In daily life many people however use "computers" or at least computing technologies, such as microprocessors and microcontrollers, without regarding them as computers. Devices such as mobile phones, personal stereos (MP3 players), TVs, and washing machines are often using computing technologies. Nevertheless users still consider their TV as a TV and do not regard them as another type of computer with a different interface.

The following vision statements offer a prediction of computing beyond the PC era. The initial and most influential statement, which also coins the term "Ubiquitous Computing", is by Mark Weiser [Weiser,91]. The vision of computing appliances, motivated more from a design and human interface perspective, by Don Norman [Norman,98] shows further issues on the theme of "invisible computing". A human centric view of computers that disappear and the implications on technologies is the focus of *European Disappearing Computer* initiative [Wejchert,00], which was published 10 years after Weiser.

2.1.1 Ubiquitous Computing

The article "The Computer for the 21st Century" by Mark Weiser published in 1991 in Scientific American [Weiser,91] became a cornerstone in the foundation of Ubiquitous Computing research.

> *"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."* [Weiser,91].

The above statement is the motivation behind the new way of thinking about computers and their use. By analysing how people interact with written information, which has become ubiquitous in industrial societies, an example of a perfectly interwoven technology is given. This level of integration can be regarded as ultimate goal for the concept of "ubiquitous computing". The central characteristics are:

- Technology that does not require active attention and
- Is ready to use at a glance.

The vision is that computers share these properties and become a part of the "natural human environment" and that they "vanish into the background". Looking at technologies which are often regarded as having these properties, such as cars and mobile phones, it can be observed that these properties are not only achieved by creating technologies but also by changing the way of life. In many cases life moulds itself to technologies.

Again comparing computing and written information Weiser states that people who can read well will not realise that they actually read something when they are looking at a street sign; instead, they just become aware of the information they are absorbing. The following statement points out a main issue of Ubiquitous Computing.

> *"Such a disappearance is a fundamental consequence not of technology, but of human psychology. Whenever people learn something sufficiently well, they cease to be aware of it. […], in essence, that only when things disappear in this way are we freed to use them without thinking and so to focus beyond them on new goals."* [Weiser,91].

Ubiquitous computing it is not just about technology and the deployment of technology in everyday environments; the human perception of technologies and the interaction with technology is the crucial test.

The concept of "Ubiquitous Computing" goes beyond having computers that can be taken everywhere and used independent of where you are. This is rather that you can take them everywhere, but you don't have to because computers are already seamlessly integrated into the world: everywhere. Ubiquitous computing is a clear contrast to the idea of virtual reality. In VR-environments technology is used to simulate the world or to create a new world, whereas

> *"ubiquitous, invisible computing is so strong that some of us use the term "embodied virtuality" to refer to the process of drawing computers out of their electronic shells. The "virtuality" of computer-readable data […] is brought into the physical world."* [Weiser,91].

The central question remains: how can computers disappear? The simple answer is that if they are cheap and small enough they will become (and in fact have become) part of other devices, they are no longer perceived as computers anymore. Furthermore, interconnecting these devices is not a goal in itself (even when it has many technologically challenging aspects), it becomes rather a prerequisite for many scenarios. Research issues where interconnecting of devices is the central issue, is also associated with the term "pervasive computing" [Burkhardt,01, p. 6].

In his statement Weiser identifies two crucial issues:

- Physical location of the usage of such devices
- The size and scale of the devices

The knowledge of physical location in later work referred to as "location awareness" or "context awareness" [Schilit,94], [Schilit,95] is a very central concept to make devices adaptive to their physical environments which seems central for interweaving the technology with its environment. The research focus in Weiser's work is to a large extent on devices that provide access to information. For this purpose they prototyped and deployed a great number of such devices [Weiser,93].

The vision goes beyond the prototypes. Individual devices do not provide something fundamentally new, however:

> *"The real power of the concept comes not from any one of these devices; it emerges from the interaction of all of them. The hundreds of processors and displays are not a "user interface" like a mouse and windows, just a pleasant and effective "place" to get things done."*

> *"[…]Like the personal computer, ubiquitous computing will enable nothing fundamentally new, but by making everything faster and easier to do, with less strain and mental gymnastics, it will transform what is apparently possible. […] But ease of use makes an enormous difference."* [Weiser,91].

2.1.2 The Invisible Computer and Computing Appliances

The concept of information appliances, as described by Don Norman in his book "the Invisible Computer" shows great similarities to the idea of "Ubiquitous Computing" [Norman,98]. The idea of information appliances is generalised to the notion of an invisible computer, which is human-centred and task-oriented. However Don Norman's motivation and approach is very different as he is coming from a design and psychology background.

> *"The proper way, I argue, is through the user-centered, human-centered, humane technology of appliances where the technology of the computer disappears behind the scenes into task-specific devices that maintain all the power without the difficulties."* [Norman,98, p. viii]

Assessing current Personal Computers he points out that they are complex, difficult to learn and use, expensive to maintain, and most notably used out of context. The following statement shows the task-oriented focus.

> *"[…] the primary motivation behind the information appliance is clear: simplicity. Design the tool to fit the task so well that the tool becomes part of the task, feeling like a natural extension of the work, a natural extension of the person."* [Norman,98, p. 52].

The transition predicted sees computers as enablers that provided value by means of the appliance in which they are embedded, rather than as devices on their own. Norman's argument and also his example 'the evolution and use of electric motors' is similar to Weiser's Ubiquitous Computing vision. The analogy refers to the provision of mechanical power. In the early days of industrialisation single steam engines were used to drive many different and distributed machines within a factory. Later motors were introduced that could be attached to various machines and devices. Nowadays we have machines and systems which incorporate a number of motors unrecognized by the user. Similarly in the information processing domain many devices only become possible because computers can be embedded. Most often these operate behind the scenes: invisible to the user.

Obviously there are always tradeoffs: having a PC that is general purpose and can be used for anything (at least theoretically) versus an information appliance that is specifically designed and built to support a specific task. To make the concept more powerful Norman specifies that appliances can communicate with each other, stating the requirements:

> *"Making a proper information appliance has two requirements: the tool must fit the task and there must be universal communication and sharing."* [Norman,98, p. 53].

But even so tradeoffs between systems that offer an all-in-on solution (PC) and information appliances remain. Norman points out that *"the tradeoff between ease of use and simplicity on the one hand and convenience on the other"* [Norman,98, p. 61] is general and well known. This however is ignoring the availability and use of context: a central argument in this thesis.

2.1.3 Disappearing Computer

The European Commission published an IST call for research proposals in 2000 in the area of future and emerging technologies named "the disappearing computer" [Wejchert,00]. The vision of the program is:

> *"A vision of the future is one in which our world of everyday objects and places becomes infused and augmented with information processing and*

> *exchange. In this vision, the technology providing these capabilities is unobtrusively merged with real world objects and places, so that in a sense it disappears into the background, taking on a role more similar to electricity - an invisible pervasive medium."* [Wejchert,00].

The main challenge is to explore a world where everyday objects are augmented with computing and communication technologies. The computer as such disappears into the background and becomes a part of an everyday object. The approach is human centric and the focus is on interaction with real objects in everyday settings. The assumption is that new functionalities and new ways of using artefacts will enrich everyday life. It is central to the vision that context matters:

> *"Artefacts will be able to adapt and change, not just in a random fashion but based on how people use and interact with them. […], resulting in an everyday world that is more 'alive' and 'deeply interconnected' "* [Wejchert,00].

The program states three central objectives that have to be addressed to advance the field, namely the creation of artefacts, the understanding of the resulting emerging functionality, and the impact on users' experience.

The creation of new artefacts – as a merger from everyday objects and information technology - is central to achieve the vision of a disappearing computer. It has far reaching consequences for computer and information systems architectures as well as fundamental implications for everyday objects. Many artefacts with communication capabilities comprise a modular information system that is deployed and used in a real world context. The hypothesis is that as these artefacts interact and exploited their usage in context, new functionality will emerge. The design and prototyping of such artefacts will provide a new dimension in peoples experience with information technology.

2.1.4 Computing in Everyday Environments – Context matters

The three visions introduced above are examples of the transition in the understanding of everyday computing. Various issues are common to all visions, most notably:

- Computers are used in non-desktop environments and mobile scenarios
- Computers become embedded into devices and real world artefacts
- The notion of operating a computer disappears: users perform a task using a device not considering it as using a computer
- A shift from expert users to non-experts is anticipated
- User experience becomes central: efficiency is not longer the sole goal

These issues have been formulated in many other research statements and article in various research communities over the last 10 years. People are approaching the problem from a computing, design, psychology, and sociology perspective. Terms such as “calm computing” [Weiser,98], “sentient computing” [Hopper,99], “pervasive computing” [Burkhardt,01], and “situated computing” [Hull,97] describe specific views on the topic. Most of the issues also relate to the transformation of computers from primary artefacts into secondary artefacts.

Assessing and analysing visions and predictions on computing, devices, infrastructure, and human interaction, it becomes apparent that:

- context is available, meaningful, and carries rich information in such environments, and
- that users’ expectations and user experience is directly related to context.
- Acquiring, representing, providing, and using **context** becomes *a* crucial enabling technology for the vision of disappearing computers in everyday environments.

2.2 The Notion of Context

The term context is widely used with very different meaning. The following definitions from dictionaries, as well as the synonyms, provide a basic understanding of the meaning of context in English.

> ***Context n 1****: discourse that surrounds a language unit and helps to determine its interpretation [syn: linguistic context, context of use] 2: the set of facts or circumstances that surround a situation or event; "the historical context"*
> *(Source: WordNet ® 1.6, http://www.cogsci.princeton.edu/~wn/)*

> ***Context****: That which surrounds, and gives meaning to, something else.*
> *(Source: The Free On-line Dictionary of Computing, http://foldoc.doc.ic.ac.uk/foldoc/)*

> ***Synonyms Context****: Circumstance, situation, phase, position, posture, attitude, place, point; terms; regime; footing, standing, status, occasion, surroundings, environment, location, dependence.*
> *(Source: http;//www.thesaurus.com)*

Also in computing and related subjects context is widely used often with different meanings. Context has a specific meaning in AI [Lieberman,00] and natural language processing (NLP) [Lenat,98] that differs to a great extent from the notion of context in operating systems and programming languages. The understanding of context in design and user interfaces engineering is again quite different. And in each of the fields mentioned there is no single definition of context. In Ubiquitous Computing the term context is central and different to the understanding of context in other fields, but no generally accepted definition has yet been established. In the following section import definitions and characterisations of context that have been published are presented.

2.2.1 Schilit: Applications Exploit the Changing Environment

In [Schilit,94] “exploiting the changing environment” is stated as one important challenge in mobile distributed computing. The term context-aware software is introduced and characterised as follows:

> *“Such context-aware software adapts according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time. A system with these capabilities can examine the computing environment and react to changes to the environment.”* [Schilit,94].

The most important aspects of context: the location, the people around, and the resources nearby, are described. On a conceptual level it is also argued that further issues, such as lighting, noise level, communication cost, and social situation are of interest and can be regarded as context.

On a more technical level, as described in the section "2.3.2 Context aware cycle" of Schilit's thesis [Schilit,95] the focus is clearly on adaptation to resources. Context-aware software is viewed as a three step process involving "Discovery: learning about entities and their characteristics", "Selection: deciding which resources to use", and "Use: employing the resource".

The prototypical system was developed using ParcTAB infrastructure and realising mainly location based services. Schilit classifies context-aware software using a 2-dimensional matrix. One dimension of context-aware software is whether an application provides information or is calling a command: each dependent on context. The other dimension is whether this action is done manually or automatically.

2.2.2 University of Kent: from Location to Context

Users are mobile and tasks often include mobility. With the development of portable computers and the advances in mobile computing, location became a parameter in such systems. A lot of work went into making location transparent systems and into the provision of location independent services. In other areas it became evident that location could enable new and interesting applications, when it is accessible to the system.

At the University of Kent the research group of Peter Brown studied the use of mobile computing systems in field work scenarios [Pascoe,99]. Various applications that use mobile devices that are made aware of their location have been developed [Brown,96], [Brown,98b], [Pascoe,98], [Pascoe,98a]. Their understanding of context-awareness is reflected in the following citation:

> *"[…] 'context awareness', a term that describes the ability of the computer to sense and act upon information about its environment, such as location, time, temperature or user identity. This information can be used not only to tag information as it is collected in the field, but also to enable selective responses*

> *such as triggering alarms or retrieving information relevant to the task at hand. Because of the importance of location in fieldwork applications, the hand-held computers used in the project are normally connected to a GPS receiver. Other environmental sensors could, of course, be added if required."* [Ryan,98].

The focus of their work is on handheld mobile devices, which are used for information collection and information retrieval. Nearly all applications mentioned are based exclusively on the use of location, acquired using a GPS unit. A typical application of this category is 'stick-e-notes' [Brown,96]. A stick-e-note consists of a context and a body. If the context described in the note is matched by the readings from sensors in the real world the body is triggered. In particular the description of context includes location, the direction the device is facing, and the time. The body can contain information or a script that is executed when triggered [Brown,97]. A whole application or document consists of many notes that are bundled together.

In their research they made an interesting observation that context aware applications can be discrete or continuous. The stick-e-notes concept is an example for a discrete application; at certain well defined points, actions are triggered. In contrast, continuous applications always take context into account and continuously update the parts that are dependent on context [Brown,97].

Even as their practical focus on applications is very much on location and in particular outdoor location using GPS, their model sees context as a nearly unlimited concept.

> *"Indeed you could argue that every application which takes some account of the user is a context-aware application. In practice, the adjective "context-aware" is attached to applications that are mainly driven by the user's context. They tend to be mobile applications […]."* [Brown,97].

A narrower definition was then given in [Pascoe,98], where context is defined as a subset of physical and conceptual states that are of interest to a particular entity. It is interesting to observe that the argument is for the "user's context". Here implicitly a connection between the sensor and the user is drawn. This is further explored when discussing the distributed nature of context, see chapter 6.

2.2.3 Lancaster University: Guide Project

The work carried out in the GUIDE project at Lancaster University [GUIDE,01] focused on how context can be used to advance a mobile information system for visitors to a historic town [Cheverst,98], [Cheverst,00], [Davies,98]. The following statement from Keith Mitchell's thesis outlines their understanding of context [Mitchell,02]:

> *"[…] two classes of context were identified, namely personal and environmental context. […]. Examples of environmental context include: the time of day, the opening times of attractions and the current weather forecast."*

The discrimination in personal and environmental context is quite interesting. Most of the issues that are classified as personal context are often also referred to as user profiles, especially in the domain of web information systems and artificial intelligence. Usually these matters (e.g. the user's interest or budget constraints) do stay the same while the application is used (e.g. one walk around the city). Location is also in this category, modelled as discrete variable that is changing very frequently while running the application. These contexts are directly related to a particular user; in contrast the environmental contexts are of a more general nature [Davies,01].

All context location is based on non-physical sensors, or in more general terms: information. This is either specific to a particular user (e.g. the users preferences or profile) or it is generic like information available on the World Wide Web (e.g. opening hours or weather forecast).

In the GUIDE project the main objective when using context was to present information in a suitable way for the context that the user is in, at any particular time.

2.2.4 Anind Dey: Supporting Context-Awareness

Anind Dey worked on the provision of architectural support for context-aware applications and provided the following general definition of context in his thesis [Dey,00]:

> *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves."*

The central point of this application-centric definition is that information that describes the situation of an entity is context. This definition clearly states that context is always bound to an entity. This is similar to the approach followed in this thesis, as outlined in chapter 4. The entity itself is regarded as something that is relevant to the interaction between a user and an application. The user-application relationship is rooted in the traditional notion of an application, but not limited to it.

By considering the application, and implicitly the state of the application, as an entity that is characterised by context a feedback loop is introduced. A change in the application will inevitably lead to a change in the context, perhaps as reaction to a changing situation. For certain types of applications and scenarios this is an elegant way of influencing context, however in other domains this cycle may increase the complexity of the model and eventually of the implementation.

The following definition for context-aware systems introduces the user's task as a concept, which is itself context (as it characterises the user's situation) but it is also used to determine the relevance of information and services. This illustrates that context can hardly be seen in isolation.

> *"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."* [Dey,00, p.6].

The application focus of systems that are considered context-aware are on systems that provide information and services dependent on context. In their work they also include applications that capture and tag information in the background and also systems where context is used to customise the interaction and the interface, while always providing the same information.

2.2.5 A Semi-Formal Approach to Context

In [Crowley,02] the assumption is made that user's actions are generally goal driven. It is acknowledge that most often actions are related to a number of goals that are pursued simultaneously rather than to a single well-defined goal. This is extended to satisfy the notion of invisible computing where explicitly interacting with the computer is usually is not regarded as a goal of the user. The term activity is introduced to model the observation that the user is concerned with several tasks at once. Also tasks that are not directly within the current activity are included and refereed to as background tasks. Altogether this is considered as the user's context. In their argument they make a further distinction between the user's context and the system's context:

> *"The system's context is composed of a model of the user's context plus a model of its own internal context. The system's model of the user's context provides the means to determine what to observe and how to interpret the observations. The system's model of its own context provides a means to compose the federation of components that observe the user's context." [Crowley,02]*

A general approach for recognition systems is assumed, where sensors provide low level information which is then available as observable variables which are numeric or symbolic. The data generated may be synchronous or discrete. The basic recognition process is modelled as a series of transformations. Data or events are put into a component, influenced by control data a transformation is done which results in new data or events together with a possible change in state and capabilities. By these means a common approach from computer vision is generalised to be used in ubiquitous computing perception systems.

The formal definition of context is connected to a user and a task. A context for a particular user and a specific task implies the roles and relations that are in common to all situations, which are described by the context. The ontology is further extended and includes bottom-up components that relate to sensing and also top-down components that relate to derivation.

The approach has two major contributions: the software architecture and the conceptual model. The software architecture suggested provides means to reduce complexity by regarding calculations as a series of transformations. This also allows independence between components. The suggested model relies on a well defined user task model that is often used in classical HCI. Even so it is more difficult to find specific goals in off-desktop computing, the more general goals (e.g. Norman's pleasurably axiom, [Norman,98, p.67]) fit also into such a model. Decomposing these general goals seems for many ubiquitous computing scenarios difficult. Dependent on the domain in the decomposition process sub goals are created and then refined into activities and tasks. However when this is done successfully the effort will result in a solid model.

2.2.6 Context – A Changing Concept

Looking at the explanations for context (such as those described above) and more detailed explanations of context in PhD theses in the field [Schilit,95], [Dey,00], [Pascoe,01], [Mitchell,02] it can be observed that the notion of context evolves with the work that is carried out. As we have been exploring various fields in Ubiquitous Computing and in particular in context-acquisition, our understanding of context has also evolved and changed (see the next section).

It seems that the understanding of context is connected to the feasibility of context acquisition. Changes in the state of art in context acquisition, e.g. a new indoor location technology becomes available, influence the use and the understanding of context. In many cases this is a two step process: first a new technology becomes available and is then used in specific applications that are tailored to use this technology. Later this technology is regarded as a context provider and if necessary the conceptual model of context-aware systems is updated to include such contexts.

When looking at definitions or characterisations for context the following issues are central:

- **Scope**. What is the scope a particular notion of context covers? Is it only a certain type of context (e.g. location) or is it a general concept?

- **Separation**. Does the notion of context help to separate concerns in a given model? Separation is interesting in two ways. First, how context is interlinked with the application and second, how modular are the concepts for context acquisition and context provision.

- **Abstraction**. Is context regarded as a particular piece of information, which is tailored to fit a specific application or is it a general concept that may apply to anything?

- **Relation**. What does context relate to? Does context describe the situation of the user, the device, or the application?

A further general observation is that scope, separation, and abstraction include a trade-off. The more specific context is regarded, the easier it becomes to implement systems based on that model. However if the view is very specific it is unlikely to be helpful for many different types of application domains. Vice versa, a very general model is covering most application domains but does not provide a lot of help when realising a specific system.

The discrimination between user, device, and application becomes more important if the domain is moved away from handheld, mobile computing scenarios. In mobile computing systems where the user carries a PDA or a wearable computer running a particular application, discrimination is not necessarily required. However in Ubiquitous Computing environments where the user does interact simultaneously with many different embedded and mobile systems and also where the term application is unclear discrimination becomes important.

2.2.7 Our Understanding of Context – an Evolution

In the course of the research that was undertaken different projects that relate to context-aware computing have been carried out. In the beginning the understanding was driven from the idea that context can add a new quality to mobile computing devices and applications [Schmidt,98]. At the IMC99 workshop we proposed a tree that offers structure for context. After further developments we realised that having a such a tree is a very pragmatic approach and helpful to implement a certain class of applications, however it lacks generality. In [Schmidt,99] we published a “Working

Model for Context-Aware Mobile Computing" which is presented in the following paragraphs. This model extends the closed tree originally proposed for an open tree model that offers a general structure that can be extended to suit the application domain.

2.2.7.1 A Working Model for Context-Aware Mobile Computing

To structure the concept of context we propose the following model:

- A context describes a situation and the environment, a device or user is in.
- A context is identified by a unique name.
- For each context a set of features is relevant.

For each relevant feature a range of values is determined (implicit or explicit) by the context.

In terms of this model, a hierarchically organised feature space for context can be developed. At the top level we propose to distinguish context related to human factors in the widest sense, and context related to the physical environment. For both general categories we propose further classification into three categories each, as shown in Figure 1. We use the six categories at this level to provide a general structure for

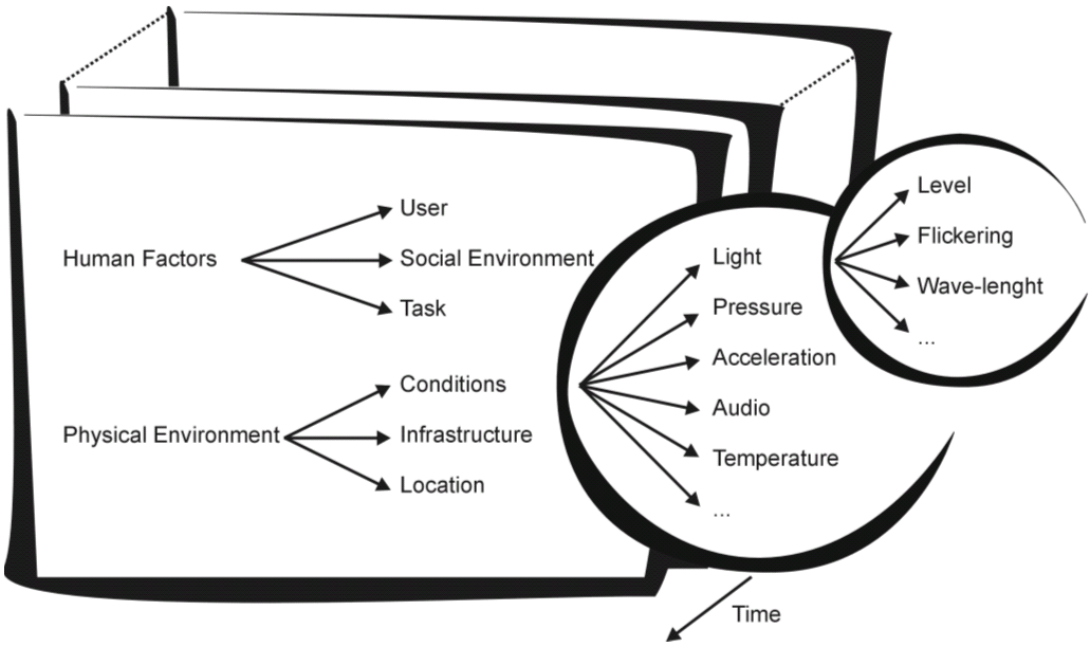


Figure 1: Context feature space.

context. Within each category, relevant features can be identified, again hierarchically, whose values determine context. Additional context is provided through history that changes in the feature space over time.

Human factors related context is structured into three categories: information on the user (knowledge of habits, emotional state, bio-physiological conditions, etc.), the user's social environment (co-location of others, social interaction, group dynamics, etc.), and the user's tasks (spontaneous activity, engaged tasks, general goals, etc.). Likewise, context related to physical environment is structured into three categories: location (absolute position, relative position, co-location, etc.), infrastructure (surrounding resources for computation, communication, task performance, etc.), and physical conditions (noise, light, pressure, etc.).

The described model provides some structure for consideration of context. For pragmatic use of context, the general challenge is to identify the set of relevant features in terms of which a situation or environment can be captured sufficiently. Situations and environments are generally characterised by a large degree of continuity over time, so that context history itself becomes an important feature for approximation of a given situation or environment. Time is implicitly captured in the history.

2.2.7.2 Reconsidering Dimensions – Project TEA

In the research project Technology for Enabled Awareness [TEA,98] we defined context as follows:

> *"Context awareness is knowledge about the user's and IT device's state, including surroundings, situation, and, to a lesser extent, location."*

To describe contexts we moved to a three-dimensional space as depicted in Figure 2, with dimensions *Environment*, *Self*, and *Activity*. A fundamental difference to earlier work was the observation that context is not necessarily related to location.

This is similar to the previous tree structure but discrimination on the top-level is different. With the introduction of a "self" dimension the issue to what context is

related to (device, user, and application) is addressed. Context has many aspects and this model is especially targeted at mobile appliances (e.g. PDAs, phones).

2.2.7.3 Revising the Model and Further Issues

Both models above have been developed with a focus on mobile systems. When developing embedded and stationary context-aware systems [Gellersen,99a], [Schmidt,02] we investigated further approaches. In chapter 4 a bottom-up approach for identifying parameters that are relevant to make things context-aware is presented in more detail.

Within the Smart-Its project [SMART,02], where the notion of collective awareness is a central research issue, it became apparent that the relation between artefacts and other artefacts, or artefacts and humans, is essential for modelling context-aware systems, however at this point such a model has still to be developed.

The following observations conclude the discussion of definitions, characterisations, and models for context, and point out a few issues to consider when re-defining context or proposing a new model:

- **Stimulate Discussions**. In many cases models or definitions have been weak or not complete, yet they have led to serious discussions of the subject. Even if

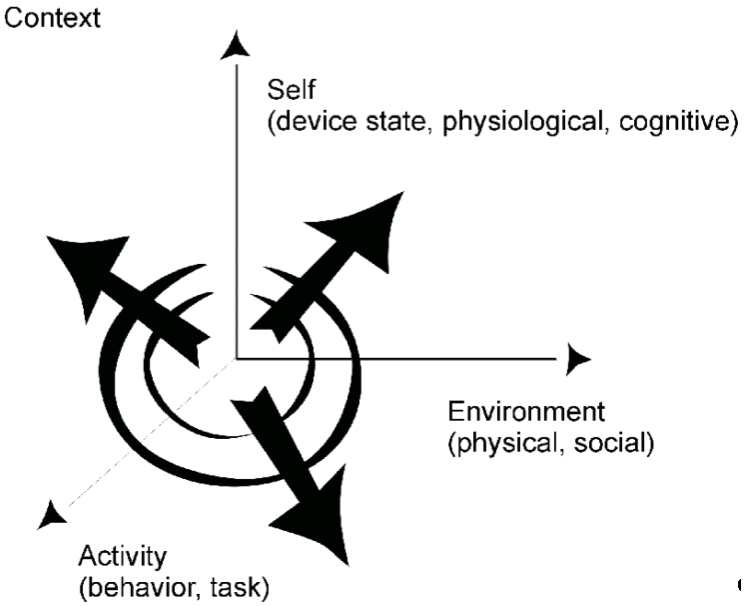


Figure 2: 3-D Context Model.

a definition or model only explains part of the issues it can lead to an interesting scientific discussion.

- **Revising Viewpoints**. Definitions should not to be considered as absolute and models can evolve or transform. When a model is not useful in a new domain or for new types of contextual information it has to be revised. This does not suggest that the "old" model was bad it's just a way of moving on and improving.

- **Scoping**. As explained earlier a very restrictive model usually helps to efficiently implement systems, whereas a general model can explain everything. Often this issue comes down to an attempt of modelling the whole world vs. modelling a specific application. The purpose of the model or definition should be used to determine the scope. In some cases it may even lead to models on different levels.

2.3 Context Related Research Initiatives and Projects

Many institutions and large scale projects have investigated the use of context; in particular location as source of information to enhance applications and systems. The following sections provide an overview of larger research efforts where context is a central concern.

2.3.1 Ubicomp Experiment at PARC

The Ubicomp experiment at PARC was the first project that investigated the idea of Ubiquitous Computing at a larger scale: in this context the term Ubiquitous Computing was coined [Weiser,91], [Want,95]. To research the implications of Ubiquitous Computing, especially in the area of networking, device technology, and human computer interaction, the ParcTab system was implemented and deployed.

The ParcTab system included different mobile and stationary devices of various sizes, in particular Tabs (small handhelds), Pads (tablet style units), and Boards (wall sized) and a Ubiquitous Computing infrastructure.

The main goals in the project were: "To design a mobile hardware device, the PARCTAB, that enables personal communication", "To design an architecture that

supports mobile computing", and "To construct context-sensitive applications that exploit this architecture" [Want,95, p. 4].

Room level location awareness was provided by the communication infrastructure that was based on infrared (IR). The main difference of a ParcTab in comparison with a standard computer is portability, communication, and context-sensitivity [Want,95, p. 22]. Many different mobile applications have been developed on the platform making use of the environment.

From the discussion and also from the summary it becomes apparent that communication and mobile use have had the greatest impact on the applications developed and on the acceptance of applications by the users. Regarding the applications that were used, context did not have a great influence at that time.

It is also interesting to note that the way the system was deployed and evaluated has now become a common method in Ubiquitous Computing research. The goal for evaluation was "*to test the entire system in an office community of about 41 people acting as both users and developers of mobile applications*." [Want,95, p. 4].

2.3.2 Sentient computing, Cambridge

The term *sentient computing* was coined at the ORL research labs in Cambridge (later the AT&L Labs) [Hopper,99]. The sentient computing project is more a conceptual framework in which many different projects are carried out, rather than a single large project. They define the term as follows.

> *"Sentient Computing: Using sensors and resource status data to maintain a model of the world which is shared between users and applications."* [AT&T,01].

The idea behind this approach is that humans observe their environment and that the interaction they carry out is directly related to these observations. To give applications similar capabilities, basically to act in context, a world model has to be provided. Humans implicitly acquire a world model that is permanently updated by observation and perception. For the system, it is suggested that a similar world model is created,

which is constantly updated by sensors. This shared perception has far reaching consequences:

> *"What could we do if computer programs could see a model of the world? By acting within the world, we would be interacting with programs via the model. It would seem to us as though the whole world were a user interface."* [AT&T,01].

The sentient computing technologies to enable sensing such models rely heavily on sensing the whereabouts of people and objects. In the early work the location of the user was at the centre of interest [Want,92], [Harter,94]. The Active Badge system provided information on the current location of people a coarse scale (e.g. mainly room level). Later work concentrated on more precise location information of objects and of people [Wart,97]. The Active Bat system, an ultrasonic location system, can provide location information of the mobile unit (called a bat) with a precession of about 3 cm. The bat can be attached to people and objects; it also can be used as an interaction device [AT&T,01]. The availability of location information allows new programming paradigms, e.g. using spatial relations as invocation mechanisms.

Within the sentient computing project various applications have been developed. Many of them are classical prototypes of location aware systems [Addlesee,01]

2.3.3 Aware Home and Further Context Research at Gatech

At Georgia institute of technology a number of projects are concerned with context and context-aware applications. The aware home initiative is a multi-disciplinary research program to investigate new classes of information technologies that can enhance the quality of life in home environments. Broadband communication, context-awareness, and personalisation are regarded as the basic building block for novel applications and technologies [Gatech,00].

Important goals in context-aware computing are stated as follows in the research proposal [Jayant,99]:

> *"The research will identify multimodal technology that robustly locates a person, and tracks activity in the short and long term, and in relation to key*

> *objects or aspects of the residential environment. Tracking technologies include arrays of smart cameras and microphones, as well as force-sensitive smart floors – all leading to an environment that is aware of its occupants and their activities."*

In the area of context-aware applications two major issues are addressed: presentation of context information to the user and tailoring the interaction according to the context and to changes in context [Kidd,99]. As the implementation of such systems is rather difficult a further goal is to provide appropriate tools and a software infrastructure to support the development.

The Context Toolkit [Salber,99] was designed and developed to allow separation of concerns for context-aware systems. In particular the insulation between the actual sensors and the application is the main objective [Dey,99]. The concept has many analogies to Graphical User Interface (GUI) systems.

Sensors are encapsulated by sensor widgets, which provide access to the sensor. As sensor systems have different characteristics compared to GUIs the architecture also takes into account that sensors are potentially distributed and unreliable. By this means context acquisition is transparent for the application. For applications often more complex contexts, the result of combining the information from several sensors, are of interest. For the aggregation of information the concept of entity servers is introduced. These entities or context servers can access context widgets and provide themselves information to applications. In cases where the translation between sensor information and context information is required interpreters are used. Interpreters are a general mechanism to translate between context values. The context toolkit was used to implement a variety of applications. The notion of context acquisition is very general in the model, basically providing a mechanism to include any sensor in the system. However the systems implemented only use a few types of sensors, mainly from the information domain [Essa,99].

A physical sensing system is described in the smart-floor project. The idea is to place floor tiles on load cells and to detect the person walking over that spot [Orr,00]. Similar work has been carried out a Cambridge [Addlesee,97] and also in sport physiology [Kistler,02]. The experiments show that the integration of load sensing

technology can provide an unobtrusive means to track and discriminate people. Discrimination is done using the force profile that is specific to a particular person. From the figures published it seems possible to discriminate a small group (typically sharing a home environment) with a very high reliability.

In the classroom 2000 project (now eClass Project) context capture in a teaching setting was investigated. A lecture theatre was equipped with display and capture technology, in particular cameras and microphones. Lectures and courses were captured, not only audio and video, but also the annotations. The captured material was then made available for access. Context was used as a significant way of structuring the material and easing access [Abowd,99].

Earlier work at Georgia Tech included a location aware visitor system called CyberGuide [Long,96], [Abowd,97] and the CyberDesk, an architecture that dynamically creates information pages based on virtual context [Dey,97].

What most of these research projects have in common is that they are evaluating in a living lab environment. Some are also evaluated with techniques normally used in HCI. The concept is similar to the one stated in the ParcTab Experiments. Researchers build systems and use them in their daily work life [Abowd,00].

2.3.4 Human Centred Computing, Project Oxygen at the MIT

The starting point of the vision for Oxygen is a critical assessment of the last 40 years of computation revealing that up to now humans adapt to machines [MIT,02]. A major criticism is that computers have not been aware of peoples needs. Their vision for the future predicts the opposite.

> *"In the future, computation will be human-centred. It will be freely available everywhere, like batteries and power sockets, or oxygen in the air we breathe. It will enter the human world, handling our goals and needs and helping us to do more while doing less."* [MIT,02].

The challenges for such a vision are characterised by the following adjectives: pervasive, embedded, nomadic, adaptable, powerful, intentional, eternal. The

explanations for most of the challenges refer directly or indirectly to the concepts of context and awareness.

The approach focuses on four technological areas embedded computational devices (E21s), handheld devices (H21s), networks (N21s), and also on adaptive software (O2S). Perception is a central issue, however the focus is mainly on vision and speech aiming to replace explicit traditional input mechanisms with conversational and gesture input.

Beyond vision and speech sensing and perception is hardly addressed. In the criticism of conventional computing and in the vision stated, awareness is a very important factor. In the approach however context-awareness is not addressed as a central issue.

2.3.5 Further Projects on Context

Further research institutes are investigating context as a central source of information in different projects. The previous sections show the variety of research and also outlines the diversity of approaches. In the following paragraphs some further projects are mentioned to round up the picture.

Including reasoning, planning, and learning in devices and artefacts, and hence embedding intelligence, is suggested by researchers at Essex University in the UK. These concepts, rooted in AI and robotics, are explored in various projects, such as iDorm (intelligent student dormitory) and e-Gadgets [eGadget]. Context sensing includes capturing explicit interaction as well as measuring environmental conditions. A central issue is how embedded intelligent agents can facilitate intelligent domestic environment [Callaghan,01]. A research focus is the development and deployment of adaptive learning algorithms, such as Incremental Synchronous Learning (ISL) [Hagras,02], which can transfer cognitive tasks from the user to artefacts and into the environment.

The adaptive house at Colorado concentrates on adaptive automation for home environments [Colorado,01]. They argue that programming home automation systems is difficult and in disproportion to the benefits gained by such a system. To "*develop a home that essentially programs itself by observing the lifestyle and desires of the inhabitants and learning to anticipate their needs*" [Mozer,99] was a main target of

the adaptive house project. Their system ACHE (Adaptive Control of Home Environments) is connected to various environmental sensors and controls lights, fans, and heaters. To acquire knowledge and to provide adaptive behaviour artificial neural networks using reinforcement learning are deployed [Mozer,98]. The system is an example how concepts from autonomous control system can be applied to Ubiquitous Computing scenarios.

The EasyLiving project at Microsoft Research is developing an architecture and technologies for Ubiquitous Computing [Microsoft,00], [Brumitt,00]. The EasyLiving concept sees three general components: the application, the UI-services which rely on UI devices, and the world model which is using sensors. The geometry of the physical world is regarded as an essential property in the system [Brumitt,00a], this is similar to other work were location was the central concern [Leonhardt,96], [Leonhardt,98]. In the area of sensing, the focus is on vision and especially with the application regarding people tracking.

The Portolano project at University of Washington within the Expeditions program is also investigating the idea of invisible computing. General goals are to '*connect the physical world to the world wide information fabric*' and '*to get computers out of the way*' [Portolano,02]. An example application area in the project is a cell biology laboratory. One of the aims is to realise invisible user interfaces based on context information such as user movement, proximity, and location. It is argued that data fusion can extract the user's intent and help to get the UI out of the way offering a lower cognitive load for the user [Portolano,99].

At TecO, based at the University of Karlsruhe, we investigated how the use of sensors can provide context from and for everyday objects [Schmidt,99c], [Schmidt,99] [Beigl,01], [Gellersen,02]. In various projects we built prototypical systems to study the implications of context acquisition and context use in everyday environments. Some of the projects are introduced in more detail in chapter 4.

In the area of wearable computing context plays a significant role [Starner,99]. Location is an obvious context of interest, but also further sensors supporting different applications are reported. Sensing physiological parameters to control a camera is investigated in [Healey,98] which is similar to work on affective wearable computers

[Picard,97]. Movements and gestures of the user are also considered as context as reported in [Randell,00], [Laerhoven,00], [Laerhoven,02].

2.4 Methodology and Evaluation

As Ubiquitous Computing is a very young field no standard research methodology is yet established. Researchers come into the field of Ubiquitous Computing often from other backgrounds, such as mobile computing, distributed systems, human computer interaction, computer vision, and psychology, each discipline having well established methods and evaluation techniques. With regard to Ubiquitous Computing and in particular to context-awareness these methods and evaluation strategies are often not optimal, covering only a subset of problems [Scholtz,01].

In Weiser's paper a case for real prototyping, and living and working with the created technology, is made:

> *"Some researchers, using themselves and their colleagues as guinea pigs… […] We have built enough liveboards to permit casual use […]. By building and using these boards, researchers start to experience and so understand a world in which computer interaction casually enhances every room."* [Weiser,91].

Building on Weiser's statement and extending the request for researchers to live with the technologies developed in their everyday environment Abowd and Mynatt propose the living lab approach.

> *"It is important in doing ubicomp research that a researcher build a compelling story, from the enduser's perspective, on how any system or infrastructure to be built will be used. […] The purpose of the compelling story is not simply to provide a demonstration vehicle for research results. It is to provide the basis for evaluating the impact of a system on the everyday life of its intended population. The best situation is to build the compelling story around activities that you are exposed to on a continuous basis. In this way, you can create a living laboratory for your work that continually motivates you to "support the story" and provides constant feedback that leads to better understanding of the use."* [Abowd,00].

This approach implies a biased exploration of results by the researchers themselves. This is in contrast to traditional evaluation techniques where independent and objective user groups are involved. In Ubiquitous Computing however there are good reasons for a living lab research method, from our experience the following are important:

- **Cutting edge technologies**. When creating new Ubiquitous Computing systems often cutting edge technologies are created. Often it is not possible to build a large number of prototypes in an early phase, as the design is improved from iteration to iteration. Furthermore these prototypes are often still fragile.

- **Multi-variant Experiments**. In an early prototyping phase when the design space is explored experiments often include changes of many variables within one experiment. To understand the implications of such experiments and to gain from the findings a deep understanding of the system is required which is often only available to the researcher. This process is not comparable to experimentation in natural science subjects, but instead is rather related to design oriented approaches.

- **Prototyping to gain Understanding**. When Ubiquitous Computing systems are created, prototyping and the use of prototypes provide insight into the technology and its implications. To prototype systems for the purpose of understanding issues and to learn about technologies, has an ongoing and long tradition in practical computer science [Engelbart,62].

- **Documentation and Training**. The ultimate goal when designing Ubiquitous Computing systems is in most cases to have devices and environments which are usable without manuals and training. In intermediate steps, when devices are still not perfected limitations on the use of such devices often apply. These limitations, which are not relevant to explore further design steps, are obvious to the researcher, but difficult or very time consuming to document.

The living laboratory research provides an efficient means experiment and to narrow down the design space with minimal expenditure. However evaluation in the real

context of use and deployment beyond the lab is essential when systems mature [Abowd,00].

The research methodology and evaluation techniques used in the research undertaken are manifold. In particular prototyping was used to explore sensing, context acquisition and interaction, as described in the following chapters. To evaluate systems and the rapid prototyping platform users and developers have been probed during informal studies that have been conducted. In chapter 8 more evaluation issues are raised and the results discussed in more detail.

2.5 Discussion

The vision of networked and embedded computing in everyday environments is referred to with a variety of terms such as Ubiquitous Computing, invisible computing or disappearing computing. These terms are widely used for the overall vision and its implication on the relationship between humans and computational devices. Other terms such a calm computing [Weiser,98] and pervasive computing [Burkhardt,01] highlight specific issues, in the first case unobtrusive interfaces and in the second extensive networking.

It is interesting to see that independent research ([Weiser,91] and [Norman,98]) from very different backgrounds (computer systems and design psychology) lead to a similar hypothesis about the future of computing in everyday life. Even though these visions and the mission statements of various projects surveyed share the notion of a human centric future relationship between humans and computers, the actual research projects are still often very much centred on technology.

To achieve a truly human centric environment that is enriched by computational devices it is inevitable that the knowledge about the world, naturally continuously perceived by humans, is also available to the system. It is interesting that the need for context is also widely shared and not disputed. Furthermore in recent years also the notion of context has been broadened and is reflected in the characterisation of context by various researchers. It is understood that context is a concept that is beyond knowing the location of objects or people. Similarly here the vision includes all types of context.

A standard research methodology and in particular a widely accepted evaluation technique of ubiquitous computing systems and context-aware systems is not yet established. Looking at projects and publications that originate from the research carried out, it is interesting to see opportunistic topic selection and evaluation that is dependant on the audience which is targeted, e.g. if a ubiquitous computing system is presented at a mobile computing conference it is positioned as mobile computing system evaluated according to the standard methods in mobile computing, if it is presented at a human factors conference the HCI issues are highlighted and evaluated according to standard HCI methods. With workshops and conference that focus on ubiquitous computing, e.g. [Gellersen,99], [Thomas,00], [Abowd,01], [Borriello,02] a move towards new evaluation methods can be observed.

2.6 Summary and Conclusions

In this chapter two original and influential visions of the future relationship between computers and people have been discussed, namely Weiser's Ubiquitous Computing vision [Weiser,91] and Norman's concept of information appliances and invisible computing [Norman,98]. To illustrate the developments since these statements the vision of the disappearing computer as formulated by the European research council has been presented [Wejchert,00]. This document also outlines which research should be carried out to reach the goals stated in the vision. Common to all statements is the observation that context: the real world around is essential to such systems.

A range of characterisations and definitions for context and context-aware systems has been surveyed and analysed. It can be observed that context-awareness and context-aware system evolved from location-awareness by generalisation. Further concepts that constitute the environment and that can be measured are included in the understanding of context. Exemplarily, several projects that reflect current trends in ubiquitous computing research and in the area of context-awareness have been presented. It can be observed that even if the notion of context is widened in most research groups, the systems that have actually been implemented rely mostly on location. Location as a prime context is very well understood [Leonhardt,98] and context-acquisition devices are available off-the-shelf, at least for outdoor use. Furthermore, the value of location as context is obvious. The value of other context

information, especially about the environment, is often not clear and measuring them often requires specific hardware.

The research surveyed lead to the following observations. Based on these observations issues to take on in the course of the research were identified:

- Most of the work on context acquisition is centred on location or shows an opportunistic sensor selection. In contrast in chapter 3 context acquisition and perception using a variety of sensing technologies is assessed systematically with respect to their potential use in ubiquitous computing systems.

- As seen from examples above context is not isolated, it is linked to entities. This fact is used to create a model where context acquisition and context use is related to artefacts. By structuring context in this way the model offers an effective means to reduce complexity. This bottom-up model is strongly connected to the approach of prototyping as shown in chapter 4.

- Related research suggests software architectures and libraries to ease the use of context when developing applications. However as there is no standard sensing infrastructure or common hardware most approaches excluded hardware or only include very specific solutions. To advance this matter a context acquisition platform consisting of hardware, software, and communication was developed and is presented in chapter 5. This is extended to a more general physical rapid prototyping system for ubiquitous computing. Together with the tools a methodology was developed, too.

- In the literature so far the distribution of context assumes an active management of such information, e.g. based on subscription models. In chapter 6 an alternative approach is suggested where context is around and available depending only based on spatial and temporal relationships.

- The very basic idea of invisible computing, interaction with a system that is not regarded as operating a computer, questions many concepts in traditional human computer interaction. An alternative implicit human computer

interaction model is presented in chapter 7. Also the notion of invisibility is investigated further.

- The research methodology that can be observed in the work reviewed in this chapter varies to a great extent. In many cases the reported evaluation is opportunistic. In chapter 8 a systematic overview of evaluation techniques and their applicability is provided.

Chapter 3

Acquiring Context using Sensors

In this chapter mechanisms to gain context information by the use of sensors are introduced. The reasoning behind this is that context-aware applications rely on the availability of information about the situation they are used in. The ultimate goal is to make available to the system, a representation of the world around that is close to the perception of the user. In this chapter it is assessed which steps are needed to provide basic perception for context-aware systems. This approach looks systematically on how to narrow the gap between the user's and the system's perception of the real world in certain situations.

As outlined earlier context sensing received little attention in ubiquitous computing research so far. In many cases context is solely based on location. For location different means of sensing and interpretation are well established. However the physical world offers a much richer environment. With respect to situation and context the contribution of further sensors, especially monitoring the physical world, is little understood.

Sensing and sensor technologies are widely used in robotics, automation, and engineering. In these environments sensors proofed to be an essential source of information to create useful systems. In general in such systems the task is well defined and specific requirements (e.g. regarding accuracy and update rate) can be determined. In contrast in Ubiquitous Computing environments sensors are often used

to monitor unstructured phenomena and new methods and techniques are required [Estrin,02].

In the following the concepts of cognition, perception, recognition, and abstraction are discussed based on the hypothesis, that context is related to parameters that can be observed when a certain situation occurs. Starting from these observations various types of sensors and algorithms are introduced and assessed for their utility for building context-aware systems. In particular sensing technologies and perception algorithms are evaluated with respect to the requirements implied by Ubiquitous Computing environments. Furthermore a layered recognition architecture is introduced that offers interfaces on various levels and supports distribution.

3.1 Perception and Cognition in Nature

In this section a short overview of perception and cognition in nature is presented as a motivation and inspiration for the further work on context acquisition. These processes in nature are extremely complex and there is no attempt in this research to recreate or simulate them. However looking at perception, cognition and the way sensory inputs are processed in nature provides many interesting ideas that can influence the design and development of Ubiquitous Computing systems.

Acting and reacting with respect to the current situation is a basic property of most intelligent systems. Looking at flora and fauna it is a major advantage in the struggle for survival to have the ability of *being adaptive*. The capability to adapt to new circumstance and situations is a vital quality for virtually all living organisms and a major advantage in the struggle for survival [Darwin,59].

Considering the evolution of life, adaptation over generations is a key success factor; this can be compared to anticipating the context of use at design time. This also relates to development processes of context-aware systems, which often occurs in generations, e.g. project TEA [TEA,98] and project Smart-Its [SMART,02]. The basic concept is to use experience from one generation for an improved design of the next generation, optimising and specialising the system. This approach is widely used [Want,02].

The use of context at run-time is comparable to the adaptation of life-forms to short term or sudden changes in their environment, such as day and night, danger, drought, and temperature changes. A prerequisite to these short-term adaptations is the availability of perception and cognition mechanisms.

Having perception and cognitive functions are the foundation of intelligent behaviour of creatures and these concepts are strongly related. The following basics of sensing, perception, and cognition are described to provide some insight in these processes. In this work it is not anticipated to model these processes for building systems, but nevertheless they are taken as motivation and as a source of inspiration for the design of the sensory part of context-aware systems.

The research into the senses that are the basis for human perception has been a challenge for a long time. A description can already be found in the works of Aristotle [Aristotle,00]. In this theory, based on the knowledge of elements at that time, it is argued that the five senses are enough to perceive everything that can be perceived. The senses named are: vision, hearing, smell, taste, and touch. This list of human senses was extended in the 19th and 20th century by: the perception of position and constellation of parts of the body, the vestibular system, and perception of pain and temperature.

Research in zoology showed that some animals have developed further senses. There are fishes and amphibians that can perceive electric fields as shown in [Scheich,86]. In [Able,90] evidence suggests that some birds have the ability to sense the direction of the magnetic field of the earth. The quality of the sense and the ability for perception varies between creatures to a great extent. In nature it can be observed that the perception capabilities of different species are closely related to their way of life. It can be observed that the requirements imposed by the environment influence the senses and perception developed by a life form.

The following senses have been a point of inspiration when searching for sensing and perception technologies[2].

- Vision
- Hearing
- Smell
- Taste
- Touch
- Temperature
- Gravity and acceleration (similar to the vestibular system)
- Position and constellation of (body) parts
- General magnetic fields and in particular the magnetic field of the earth
- Electric fields

Senses in nature cannot be directly compared to sensors in a technical world. Senses comprise the whole process from the reception of the stimulus, translation from stimulus to signal, signal transport and the processing on several levels. In particular the processing of neural signals is very complex and not fully understood. The basic assumption is that the information sensed from the environment is translated into patterns. These patterns are than associated in further process steps with meaningful concepts. This leads to the assumption that perception is not possible without memory, as claimed in [Neisser,76] and in [Goldstein,97, p. XXI].

To understand or at least interpret information that is sensed from the environment knowledge or experience is required. Creatures learn during their development how to assign meaningful and abstract situations to complex stimuli received by the sensory system. This is based on the presupposition that similar situations are characterised by similar stimuli, as discussed in detail later. Comprehension of a situation or understanding of the implications given by a situation is a further step, which is to a great extent based on the recall of experience.

[2] The perception of pain is not further investigated because they are very specific to creatures. Similarly sense that are more complex such as perception of emotions are left aside, because they are not directly related to a sensory input as they rely on multiple senses mentioned above.

3.2 Sensing Situations and Representing Context

In a very general view the concept *situation* describes the circumstances, the current conditions, and state someone is in. In this thesis the term situation will be used in this general form, based on the following definition from an observer's viewpoint.

Definition: **Situation**

> A situation is the state of the real world at a certain moment or during an interval in time at a certain location.

This general definition of situation has implications when it comes to the task of describing a situation. To fully describe the state of the real world at a given moment in time is almost impossible. Any description of a situation is therefore incomplete. A description is hence always an abstraction of the real world; the real complexity of the situation is reduced to the characteristic of the situation. The step to select the characteristic properties of a situation is always subjective. The decision which aspects are characteristic for a certain situation is left to the person who makes the description. Unstructured descriptions for the same situation made by different people will most often result in diverse descriptions. Different levels of abstraction (e.g. forced by different maximal length) of a description will inevitably lead to different reports on the same situation. A further and important issue is that a description is also dependent on the goal of the person who produces the description. In general it can be observed from daily experience that humans characterise and describe situations differently based on their role, task, goal, expectations, emotions, and knowledge. Nevertheless descriptions are the most used means in everyday life to communicate knowledge about situations, well knowing that these descriptions are neither objective nor complete.

More structured ways of describing situations are questionnaires and check-lists. Using these mechanisms the characteristic features of a situation that are of interest are predetermined by the person producing the questionnaire or check-list. To some extent the level of abstraction that is expected from the person who fills the form is determined by the space (e.g. number of lines, words expected, text box size) provided.

Matching similar situations is a task that is essential in life. By recalling similar situations creatures have means to transfer accumulated knowledge and experience from the past and gain therefore advantage. E.g. a car driver recognises a situation where there is ice on the street and proceeds slower because the driver knows from previous experience or knowledge that driving fast in this situation may be dangerous and could result in an accident. This even works if the driver has never travelled this particular road before, because for this type of situation it is not important that this is a specific road. This example explains a further challenge: the description of a situation has to be specific enough to match only relevant situations but as general as possible to match all similar and related situations where the knowledge is useful.

Definition: **Context**

A Context is identified by a name and includes a description of a type of situation by its characteristic features.

The description can constitute of a number of conditions that can be evaluated to true or false, possibly with an assigned certainty. Having such a context it can be evaluated whether or not a specific situation belongs to that context, which is described. Context is a mechanism to describe situations by their defining features and group them into one unit. In certain cases the description can consist of a single complex condition that is not necessarily human readable, e.g. an artificial neural network.

Definition: **Situation S belongs to a Context C**

A situation S belongs to a context C when all conditions in the description of C evaluate to true in a given situation S.

The following properties are desired for the description of a context:

a) All situations that are of the described type should be matched by the description
b) A situation that does not belong to the described type should not be matched
c) Given a) and b) the description should be minimal

Creating a description of a context includes similar problems to creating a query for information retrieval. To assess the quality of a description measures such a **precision**

and **recall**, well known from information retrieval can be used [Baeza-Yates,99]. Based on these definitions context can be regarded as a pattern, which can be used to match situations of the same type.

It is obvious that a situation in the real world can belong to many contexts. This is an observation that is coherent with experience in everyday life. Most often a situation does not solely belong to one type of situation. Therefore situations in the real world will be treated as belonging non-exclusively to a context. That means a certain situation can belong to none, one, or a number of contexts.

To reduce complexity it is possible to create a restricted model which only allows exclusive contexts. The set of contexts has to be selected in a way that possible situations are exclusive of each other. This makes the development of perception systems as well as context-aware applications much easier.

3.3 Sensor Data is Related to Situations and Thus to Context

A certain type of situation has characteristic features and humans recognise that a particular situation belongs to this type. This process of assessing situations is done over and over in any action humans do. It is mainly based on an implicit analysis of the sensory input from the surrounding, combined with the internal state and knowledge available. This observation leads to the following more general hypothesis.

Hypothesis 1: For all situations that belong to the same context the sensory input of the characterising features is similar.

Table 1 illustrates that assumptions can be made on specific sensory inputs, which determine the occurrence of a certain type of situation.

Context	*Related sensory input*
User sleeps	It is dark, silent, type of location is indoors, time is "night-time", user is horizontal, specific motion pattern, absolute position is stable
User is watching TV	Light level/colour is changing, certain audio level (not silent), type of location is indoors, user is mainly stationary
User is cycling	Location type is outdoors, user is sitting, and specific motion pattern of legs, absolute position is changing.

Table 1: Contexts related to sensory input.

Table 1 also shows that there are inherent limitations to the approach of finding characteristic features. Taking the first context "User sleeps" the stated related sensory input is correct for the vast majority of cases where this situation occurs. Nevertheless the "context-recall" is certainly not 100%. Looking at the following examples at least one of the characteristic features is not true: someone sleeps in a sleeper during a train journey, a home less person sleeping in the street, and a nightshift worker sleeping during the day. As humans we still would consider these situations belonging to the context "user sleeps", but the further context-recall is pushed to an optimum, the more complex the descriptions become and the closer we get to modelling the world. Also examples can be created where "context-precision" is not at 100% either; consider a situation where a detective is doing an observation, laying on the floor in a dark room watching someone.

If systems should be enabled to make use of context there is a need for providing access to a sensory system. A prerequisite to find out whether or not a specific situation belongs to a context is to have information sources, such as sensors, that deliver the sensory input. Knowing the values of the characteristic features at a specific time or over an interval makes it feasible to determine if the situation belongs to a context or not.

To verify the hypothesis sensor data collected in various situations that belong to a certain context as well as situations that do not belong to the context has to be analysed. For the selected characteristic features there has to be a similarity between all the situations that belong to the context, whereas for the situations that don't belong to the context there has to be significant dissimilarity. Based on the selection of sensory systems and characteristic features it can be decided whether or not the hypothesis holds. When creating systems this is usually an iterative and constructive process, rather than just testing the conditions. Data mining techniques have been successfully used in our projects to find characteristic features. See chapter 5 for a detailed discussion of a method to build systems that acquire and use context.

The remainder of this chapter will introduce the topic of sensing in Ubiquitous Computing scenarios. In particular guidance for the selection of sensory systems, feature processing algorithms and architectures will be given. These insights should help to choose the *right* sensors that represent the important issues of the situation and

also are compliant with the requirements imposed by this type of situation. Furthermore it will also offer help for choosing the *right* algorithms and architectures for the usage of context.

A general problem is to find the borders of a context, situations all belonging to one context are still different.

- on the non characteristic features they vary to a great extent
- within the characteristic features they vary to some extent

Algorithms that are built to match contexts from sensory input must take this into account, in particular allowing variance in sensor data, because situations are never exactly the same.

3.4 Requirements on Sensing in a Ubiquitous Computing Environment

When developing, designing and building devices that facilitate context-awareness the technical and economic constraints have to be taken into consideration; see Table 2 for an overview. Technical constraints are related to the type of devices or system that should be enhanced by context-awareness, economic constraints are mainly bound to the cost of a device, the system or service, and social constraints are based on the anticipated users. Most of the issues are of particular interest for mobile devices but many of them are also important when building embedded or stationary devices and systems. Using sensors in Ubiquitous Computing environments to gain context and to facilitate new ways of human computer interaction requirements are very different from robotics where sensing is also widely used [Siegert,96,p.28].

Requirements on Sensing in a Ubiquitous Computing
Design and Usability
Energy Consumption
Calibration
Start-up Time
Robustness and Reliability
Portability, Size and Weight
Unobtrusiveness, Social Acceptance and User Concern
Price and Introduced Cost
Precision and Openness

Table 2: Constraints on Sensing.

The following criteria give guidelines as to what questions should be considered when selecting sensors and algorithms. These issues are not isolated. Most often there is also a trade-off between certain issues discussed here and the added value provided by a sensor or algorithm.

3.4.1 Design and Usability

Introducing sensing technology in devices can have severe consequences on design and usability. It becomes a major challenge to include sensors into the design without compromising aesthetics and usability. In particular if sensors need a window to the real world (e.g. light sensors and gas detector) this becomes a major issue. Also if sensors require a certain user interaction, e.g. touch sensors that can tell if the user holds the device in his hands, they become an integral part of the design of the system. Sensors that are directed or where a certain basic position during operation is assumed (e.g. gyroscope) must be integrated into the design so that these requirements are met.

When constraining the way people can use a device because of the sensing technology that is build into the device there has to be a convincing added value that comes from sensing. More general the utility and usability of a device should not be compromised by the inclusion of sensing technology.

3.4.2 Energy Consumption

For many mobile devices, especially in the consumer sector, battery life time is a crucial aspect. For mobile phones, PDAs, and wearable computers the operating time before they have to be recharged is an important discriminating factor. This class of devices is however very much suited to be enhanced using context, because they are used in different situations [Schmidt,98]. When sensing and perception is build into such devices the additional power consumption becomes a major issue. Therefore it is of great importance to design the perception module with the goal of minimising power consumption. This includes decisions on which sensors are used, what processing hardware and algorithms are deployed and how often sensing is invoked. To argue in favour of sacrificing battery power for sensing and perception the added value for the device, application, and ultimately for the user, must be significant.

A goal should be to add perception with a positive power balance. Here the basic idea is that due to the availability of context information power saving on the main device can be more effective and the energy saved here is more than the energy spent on the perception module. An example is to switch on a PDAs screen only when it is touched by the user, as we have shown in [Schmidt,00a]; similar work for a watch is reported in [Cakmakci,02].

3.4.3 Calibration

Sensors most often involve complex electronic, mechanical, or chemical processes to measure certain environmental conditions. To get precise sensor data it is often necessary to calibrate sensors before use. If this can be done during production of systems using such sensors introduces extra costs. When it has to be done in the user's environment it makes the device much more complicated to deploy. When calibration has to be performed more often (e.g. each time before using the device) this will impose a negative effect on the usability. Especially sensors that need specific calibration procedure or have a tendency to de-calibrate over time are difficult to use in a consumer market.

To minimise the need for calibration or to avoid it at all appropriate post-processing mechanisms and algorithms can be used on certain types of sensors. One general approach is to work on relative values, changes in reading, and derivatives rather than on absolute values of sensors wherever possible.

3.4.4 Start-up Time

The time needed before a device is operational is critical for certain applications. Especially where an instant-on feature is required additional setup-time or boot-up time introduced by the perception module is not acceptable. For the selection of sensors and algorithms these are important criteria. Therefore sensors that need a warm-up time (e.g. location sensor via GPS, gas sensing) are only useful when the delay is accepted by the user, or if it is of no particular interest to the application.

Methods and algorithms can take this into account and can be adapted to support instant-on behaviour. Consider a perception module that provides information about the audio environment (e.g. noise level, type of noise, number of speakers, etc).

Typically the device will give this information by analysing a time interval (e.g. the last 20 seconds). This would then lead to a start-up time of the module, in the example at least 20 seconds. To overcome this problem different approaches can be used, such as using the information that was valid before the device was switch off, calculating a rough estimate of the values in a much shorter time when the system is booted (e.g. just looking a two seconds of audio), or just providing a default value until calculated values are available.

3.4.5 Robustness and Reliability

To make devices useful in everyday environments they have to be robust. Many sensors and sensing devices are designed to be used in a laboratory environment with great care, how people use consumer devices is however completely different. For example, it is not expected that a mobile phone requires servicing for re-calibration after it was dropped it from a table, whereas for a scale in the lab this would be perfectly fine. Sensors also impose constraints on the way devices are designed, e.g. if the device should be able to pick up gas concentrations in the environment it is extremely difficult to make the device water-proof. Building windows for the sensors into devices can reduce the robustness.

Many sensors are very sensitive and fragile; therefore it is difficult to integrate them into everyday devices to work reliably over the lifetime of the artefact. When selecting sensors this should be taken into account. The algorithms selected should counter the effect of unreliable sensor information or at least be designed to cope with it.

3.4.6 Portability, Size and Weight

Adding sensors and processing for perception will usually add to the size and weight of an artefact. For mobile devices in particular the increase in size and weight has to be considered because it can easily lead to decreased portability. This is especially important for mobile artefacts as decreased portability can easily result in a worsened usability.

Size and weight are not only connected to sensors but also indirectly to the perception algorithms used. More complex and computing intense algorithms may need more

processing power and storage. This leads to a greater demand on the processor and can ultimately lead to higher power consumption and hence to a heavier and bigger device, due to the batteries required.

3.4.7 Unobtrusiveness, Social Acceptance and User Concern

By the introduction of sensors the appearance of a device or environment can be changed dramatically. Building sensing unobtrusively into artefacts and environments is a goal that is shared in many Ubiquitous Computing projects and is also related to the concepts of calm technology as introduced in [Weiser,98] and ambient interfaces [Wisneski,98], [Gellersen,99b].

When considering the inclusion of sensing technology it is important to assess user concerns about the sensors that should be used and their appearance in the design. Especially when devices are equipped with complex optical or acoustic sensing capabilities, users are often concerned about their privacy [Mann,01].

3.4.8 Price and Introduced Cost

Devices become more complex and also additional components are needed for built-in perception. The additional cost introduced by sensors, additional processing power, development cost, and potentially increased maintenance costs have to be related to additional benefit that an application gains by being context-aware.

Calculating the benefit of having context information available vs. the introduced cost is not easy. Often it is highly subjective because it is dependent on the value that people assign to the additional functionality enabled through context. Moreover many variables affect each other so it is more of a design decision than a calculation.

Nevertheless given that certain context information should be available in the device or the environment, price and introduced cost can be used to assess what sensing and perception option is the most useful.

3.4.9 Precision and Openness

In traditional sensing systems the require precision of the sensors can be determined by the application that is supported. In a closed system requirements and conditions can be matched by a specific sensing system. When creating a Ubiquitous Computing

environment that can support many different applications and should be open to new developments, the requirements and conditions can not be clearly determined upfront.

In many cases the selected precision is a trade-off with other requirements. It is therefore useful to select sensing systems so that the information offered is sufficiently precise to support anticipate applications while keeping the constraints on other requirements minimal. Creating a system that is open to new sensors and perception methods will further ensure that new applications with different requirements can still be realised.

3.5 Sensing Technologies and Systems for Data Capture

Sensors and sensing technologies are widely applied in robotics, automation, and process control. In the field of sensor technology major advances have taken place resulting in significant improvements with respect to physical size and weight, power consumption, processing requirements, interfacing options, reliability, robustness, and price [Göpel,95], [Saffo,97], [Sensor,99], [Baltes,01], [Sensor,01]. These developments suggest that a variety of sensors are useful and deployable in Ubiquitous Computing environments to provide information about the real world. An overview of technologies considered in more detail in this chapter is presented in Table 3.

The range of sensors available is large, e.g. the nomenclature of the ‘sensor 99’ fair lists more than 130 categories of sensors [Sensor,99]. However many of these sensors are very specific to certain applications or they impose very specific requirements for

Sensing Technologies
Light and Vision
Audio
Movement and Acceleration
Location and Position
Magnetic Field and Orientation
Proximity, Touch and User Interaction
Temperature, Humidity and Air Pressure
Weight
Motion Detection
Gas-Sensors and Electronic Noses
Bio-Sensors
Zero-Power Sensors

Table 3: Technologies for context acquisition.

their operation so that their application in a Ubiquitous Computing environment is not practicable.

Within the research carried out, many different sensors have been examined and evaluated with respect to their applicability for Ubiquitous Computing applications. The survey presented here will provide the reader with a selection of sensors and sensing systems that were most useful.

3.5.1 Light and Vision

Single optical sensors (photo-diode, colour sensor, IR and UV-sensor, etc.) can be used to gain information on the light intensity, the density, the reflection, the colour temperature of the light (wavelength), and the type of light (sunlight, type of artificial light, etc). Different light sensors are available that are receptive to a specific wavelength or a specific spectrum of light (UV, IR, or human eye like) [TAOS,02].

Light sensors are a source of rich information at a very low cost. Energy consumption is fairly low (e.g. TSL250 1mA at 5V); most sensors are simple to interface (e.g. analog output, pulse-width output) to a microcontroller, these sensors are also mostly robust and cheap.

Beyond the immediate features further information can be deduced when monitoring the signal over time. From patterns in the light (e.g. 50Hz flickering or light emitted by a TV) cues about the environment can be calculated [Schmidt,99a].

Combining more light sensors on one device (e.g. 2 on the lower side of a PDA, one at the front, one at the back, and 2 on the top) information about the light distribution can be used to reason on movement and further contexts (e.g. direct light, indirect light, device placed on a surface, held by the user, etc).

C-MOS Camera-modules can be used similar to an arrangement of light sensors by using lenses and appropriate algorithms**.** Beyond this, cameras offer a wide spectrum of information that can be sensed, such as visual information about the environment, which can be obtained with little processing power (e.g. main colour, motion) or richer contexts that need more processing power (detection of objects, landmarks, people, gestures etc). Many algorithms are available to gain further information such

as a colour histogram, recognition of shapes, markers and objects, and motion tracking.

Camera modules have become inexpensive but for most applications they have higher demands on processing and storage that cannot be met by low end microcontrollers. Increasingly digital camera modules that have processing power already built-in become available. When cameras are used, the direction that they are facing becomes a major design issue. In contrast to light sensors people often feel uncomfortable when cameras are around.

3.5.2 Audio

The audio environment is a further rich source of information; humans use audio as one main communication media. The information available from paying attention to audio is manifold, reaching from simple features such as volume and spectrum to full-fledged audio processing including speech recognition.

Microphones and amplifiers can be used to capture the audio in the environment. Microphones are available for different frequency spectra. By these means and the usage of band pass filters the detection of audio can be restricted to certain frequencies.

The output of the amplifier can then be attached to an analog input of a microcontroller or to an analog-to-digital converter. Depending on the features that are of interest sampling rates between 4 kHz and 100 kHz with 4 to 16 bit accuracy are useful. For gathering the audio data the Nyquist theorem applies; the sampling rate has to be at least twice the frequency of the maximum frequency that is required to be sampled. In some case, e.g. when only the noise level is of interest, lower sampling rates over a longer time may be used and than the values are estimated based on statistical measures. For specific features such as volume and spectrum specific Integrated Circuits (ICs) providing these functions in hardware are available, e.g. [National,02].

A variety of further features (e.g. based frequency) can be calculated even on minimal hardware. As we have shown in the project TEA it is feasible to discriminate different

types of audio input (noise, music, speech) on a microcontroller using little resources [TEA,98].

Using multiple microphones that are interconnected can provide information about the location of an audio source [Svaizer,97]. Having a number of spatially distributed microphones (e.g. on each corner of a device or in the environment) identifying context that is related to distributed sound sources becomes possible [Schmidt,99b]. Correlating the audio streams from different distributed microphones is a computationally cheap way to acquire more information about the environment [Schiele,01]. Using more powerful systems speech analysis can by carried out.

A option is to sense audio that is beyond human perception, for example the use of ultrasonic sensors to augment human sensory capabilities.

3.5.3 Movement and Acceleration

Acceleration and patterns of movement are very valuable as information when considering mobile and wearable artefacts. Knowing whether or not devices have been moved and in what way they are handled can become a helpful source of context information. A basic constraint in the real world is that creatures and artefacts do not change their whereabouts other than by movement and acceleration.

When humans interact with other humans it most often involves some act of movement of a certain body part. Also when humans interact with artefacts in their environment this is often connected to some movement or acceleration of the artefact.

Sensing these movements and accelerations can be realised in various ways and at various costs. Exemplarily the following three options: motion switches, accelerometers, and gyroscopes, are assessed further.

Motion switches offer a simple form for detecting movement. Usually they are mounted to a device in a way that the anticipated movement will result in change of the position of the conductive element in the switch. This will then lead to a change in the state of the switch (e.g. from open to closed or vice versa) and this can be exploited when connected to an interrupt line or a digital input of a system. These

sensors offer a fairly simple mechanism to wake-up microcontrollers when they are in an energy saving sleep mode.

Accelerometers are available as integrated micro-machined devices combined with driving electronics in an IC, e.g. the ADXL202E [Analog,01]. Accelerometers provide information on the acceleration, but the acceleration output is dependant on the orientation of the devices. This has the advantage that accelerometers can be used to measure the orientation of a device with a great accuracy if the device is not moved. The drawback is however, that the absolute acceleration value is less useful if there is no knowledge about the orientation. This is especially true for mobile and handheld devices where it is a major issue. One solution is to look at relative values rather than absolute ones. These sensors are fairly easy to interface to a microcontroller (analog or pulse width) and their power consumption is rather small (e.g. ADXL202E 0.6mA at 3V). Also the device size is minimal. The changes in acceleration are reflected quickly in the sensors output, in the order of milliseconds.

Using Gyroscopes is another option when angular velocity is of interest. These devices are generally more expensive, bigger in size, and also need more power [Murata,99]. They usually supply an analog signal that represents the represents the angular velocity in volt per degree per second.

For many applications, especially when no prior knowledge about the orientation of the device is available, it can be very useful to combine three accelerometers or three gyroscopes to gain information about acceleration in all dimensions [Sato,01].

3.5.4 Location and Position

Location and position have been widely investigated for their use in context-aware systems, as evident from many projects. When discussing location and position, issues such as co-location and proximity are also relevant.

For sensing location outdoors GPS and dGPS are most popular and easy to use [Hofmann,97], [Letham,01]. However GPS has a long boot-up time (typically 30 seconds to one minute). This time can be reduced by using further information gained earlier or over a data connection. The output of a GPS location system provides the position of the device. Depending on the number of satellites which are visible the

accuracy is within a few meters and with dGPS in the region of centimetres. Most devices support the standardised NMEA0183-format that is exchanged over a serial line protocol. Most GPS systems offer additionally a propriety protocol that is more powerful.

Another option is to use information provided by a cellular network, such as the information about GSM base stations in range and their link quality or just the GSM-cell booked in. The location information is usually only supplied to services in the network [Nakanishi,00], [Park,02]. On some GSM modules or phones the information about the base station booked in and the signal strength can be read out and captured over a serial line protocol (e.g. debug mode of an Ericsson PF768 phone).

In certain setups radio beacons can also be used to provide location information. Here both specific hardware and software is used or the location system is built on top of an exiting infrastructure (e.g. as in GUIDE using WaveLAN [Cheverst,00a]). Further issues on outdoor location systems are discussed in [Bulusu,00].

Another method is to look at what TV and radio stations can be received (e.g. using Radio Data System – RDS), on which frequencies and with what signal strength. This can be realised using a radio and/or TV module where this information is accessible. The location is then determined by matching the reception patterns with reception patterns of which the location is known. There are no commercial products available using this technique but it seems to be an interesting option for systems that include a radio or TV receiver, such as car radios or TV sets.

Comparing these methods GPS offers the service to find the location anonymously, whereas in the other systems it is very much a design decision whether or not a central system is involved in finding the position. The accuracy of the position gained from outdoor location systems can be ranging from town level down to centimetre level.

In indoor scenarios where coverage of a building is required different technologies are available and deployable, in particular IR-beacon systems [Butz,00], RFId-Tags, and ultrasonic location systems [Ward,97], [Nissanka,00]. Further approaches are researched, such as WaveLan triangulation, location systems based on existing infrastructure, and RF-beacons [Bahl,00], [Small,00]. When setting up such systems

the design decision can be made as to whether or not anonymous location finding is supported. This usually comes down to the question whether the location is calculated on the local device or within the network. The accuracy of indoor systems is very much dependent on the environment [Regenstein,01].

In our experiments and from reports of other researchers it appears that most indoor location approaches provide very high accuracy under laboratory conditions, but still perform poorly in real environments where people walk around, doors are opened and closed, laptops and PDAs use WaveLAN and Bluetooth and where CRT-screens are switch on and off. Exceptions are the AT&T active bat system [Wart,97] and the MIT cricket location system [Nissanka,00] which are not commercially available.

In order to sense co-location the approach of using radio beacons that can adjust their outgoing signal, or measure the strength of the incoming signal, can be taken. By making the communication range adjustable the degree of co-location can be selected. As shown in the Smart-Its project [SMART,02] this can give an accuracy of about a metre most of the time, however changes in the environment introduce significant errors. Co-location between devices and the environment can also be realised using RFId technology with long range readers. The use of strong readers and large antennas is very disputable in an environment where people live or work.

Location sensing is somehow different from the sensors discussed earlier. The aim of most location systems is to offer a ‘sensor’ that gives meaningful symbolic or geometric location information. This information is then most often used as a trigger, or an index to access further information.

3.5.5 Magnetic Field and Orientation

Different types of sensors are available to detect magnetic fields. Some are designed to detect the earth’s magnetic field whereas others are constructed to detect the proximity or change of a generated magnetic field. Hall-sensors detect the flux of the magnetic field applied.

Sensors that detect the earth’s magnetic field are the basic building blocks for an electronic compass. The output is related to the direction of the magnetic filed and can be used to figure out which direction is north. These sensors are also available

combined with electronic circuits in one component that provide this information on a higher level [Honeywell,02], [Philips,02]. Advanced modules offer information similar to a compass, so the direction of a device or of a movement can be determined.

In our experiments we realised that in modern environments (e.g. offices with computers monitors) this sensor can give false information. Nevertheless there are many application areas where the orientation is of significant value.

3.5.6 Proximity, Touch and User Interaction

Similar to the argument introduced with movement and acceleration it is also a basic observation of the real world that people often interact with things by touching them. Sensing that a user interacts with an artefact can be implemented in various ways and also with different levels of detail, ranging from the mere fact that the user touched the device, to the way the artefact is hold.

Before the user can touch an artefact they have to come close to it; therefore sensing the proximity can offer information that user interaction is likely to be ahead. A very simple way to sense proximity is to use a capacitive sensor. Such capacitive sensors are available off-the-shelf and offer a digital signal when approached and a threshold is crossed. For larger settings or to integrate them into artefacts, such capacitive sensors can also be built into objects using metal sheets and a driving electronic circuit. Proximity sensors that offer the distance of an object or the user hand are also available based on light. The analog output is related to the actual distance of an object in front of the sensor.

Humidity sensors can also be used to provide information on the proximity of users. The humidity rises when users approach an artefact. This rise however is extremely small, and high quality sensors are required to measure that change.

Conductive surfaces on artefacts can be used to get information on touch [Hinckley,99]. These surfaces can also be used to measure the skin conductance of the user and to some extent muscle tension. There are additional amplifiers required to provide the signal in a way that it can be read by a microcontroller. The values on skin

conductance and muscle tension are also dependent on how strong the users grip on the artefact is, because the measurement also includes the transition resistance.

A further option to measure user interaction and touch is force sensitive resistors. These surfaces change their resistance according to the force applied to them. Putting them on the surface of an object, means that the object can be use to measure how strong the grip of the user is. Using strain gauges on the structures of artefacts can also provide information about the way a device is held. By the user's grip the artefact is minimally deformed and that can be measured, as our experiments with a ball pen showed.

Other sensors such as light sensors, temperature sensors, and CMOS-cameras can also be used as the source for information about touch and proximity. Typically light sensors are covered when a user holds the device and when held for longer the device also heats up from the body heat.

Acquiring information about touch and proximity can be utilised to realise when the device is being used or in the case of proximity to anticipate that the device will be operated in the near future. By these means these sensors can reduce energy consumption significantly, especially for devices that only need to be operational in the user's hand. Touch and the way artefacts are gripped can also offer further information about the interaction process and in some cases about the user's emotional state.

3.5.7 Temperature, Humidity and Air Pressure

Temperature can be sensed using extremely simple thermal resistors or more sophisticated temperature sensors with built-in driving circuits. Such temperature sensors are available with analog or digital interfaces and offer high accuracy. A rough knowledge of the temperature can be used to help identifying the type of environment the device is in, often the temperature can be used to rule out a certain condition rather than to indicate one. E.g. given a temperature reading of -10°C can be used to rule out with a very high probability that the device is used indoors at that moment. Whereas a reading of +20°C (room temperature) does not strongly indicate that someone is indoors because this temperature appears quite often outside, too.

Using high accuracy temperature measuring (e.g. resolution of 0.1°C) can help to find transitions between situations, indications on the usage patterns, and also changes in whereabouts of an artefact. For example having temperature sensors on various artefacts and also in the environment can help to determine co-location of artefacts. Looking at the time-stamped temperature history of objects co-location at certain periods can be assumed or ruled out.

When building applications that are operated in environments where temperature has an important impact, for instance, for fire fighters, in arctic regions, cold storage rooms, or desert environments the information can be of great value at little cost.

Sensors for humidity are slightly more complex and also more expensive than sensors for temperature. Humidity sensors are available based on capacitive and resistive technologies and also as modules that provide an analog output that is proportional to the humidity level. Humidity sensors react slowly to changes, in the order of seconds. Beyond applying humidity sensors for measuring whether conditions they also can be used to get information about transitions and changes. People that are in spaces also change the humidity, which can then be measured, e.g. when people are entering a room the humidity will increase.

Absolute air pressure gives an indication on the altitude and it can also be used as a barometer. Detecting changes in air pressure can indicate certain actions, e.g. a closing door in a room or vehicle will change the pressure minimally, similar changes happen when driving trough a tunnel.

3.5.8 Weight

Weight is an intrinsic property of all objects and creatures. The straightforward way of measuring weight of objects and creatures unobtrusively is to create environments that have load sensing technology built in.

Load cells are sensors that can be used to measure weight. These types of sensors are widely deployed in industrial systems and also commonly used in electronic scales. Load cells can be manufactured to measure loads on nearly any scale, ranging from measuring ingredients for pharmaceutical productions in milligrams, to the weight of a freight train with several hundred tons. The resolution is dependent on the range and

also on the quality of the device. Load cells are available based on different technologies and also with different interfaces. Commonly used industrial load cells, based on resistive technologies incorporate a Wheatstone-Bridge, and provide an analog output signal. Without signal conditioning they typically provide an output range from 0 to 20mV. This voltage has to be amplified and can then read into the microcontroller via an analog-to-digital converter. Off-the-shelf load cells are available in different physical shapes and sizes (e.g. low profile, beam, and S-beam). Depending on the object that is going to be equipped with load sensing capabilities the appropriate one can be selected. If specific measurements are required strain gauges can be used to measure deformation of structures and so load sensing can be directly integrated.

The obvious quantity to measure is the absolute weight of the objects as a discriminating property, as used in [Konomi,99]. This is using the basic scale functionality. Analysing the change of load when an action occurs can reveal further information, such as who is walking over a floor tile, e.g. as described in [Addlesee,97] and [Orr,00]. In the experiments described in chapter 4 and published in [Schmidt,02] and [Schmidt,02a] we show that distribution of weight, hierarchies of load cell arrangements and analysis of the events is beneficial to build context-aware systems.

3.5.9 Motion Detection

Detecting and observing the presents of people in a space is interesting to many applications in Ubiquitous Computing. Detecting motion of subjects and objects in a certain space can be facilitated using different technology.

A common way for motion sensing is the use of Passive Infrared Sensors (PIR). These sensors detect changes in the heat flow in the environment and can therefore detect humans and animals moving in the detector region of the sensor. PIR sensors are available with analog output detecting a moving heat source. As modules with an additional driving circuit they are also available providing digital output offering the binary information whether or not someone entered or left the detector region. These sensors always have a directed input and are available with different lenses offering observation angels of 30° and 180°, and ranges of 2 to 15 meters.

In stationary devices and shared environments, motion sensing can offer great value at minimal cost. In mobile devices, motion sensing is more difficult because when the device is moved, it is hard for the sensor to tell if something around it is moving or if the device itself is in motion. However, when the mobile artefact can also sense movement this conflict can be resolved.

Motion detection using PIR sensors, compared to video analyses, is less powerful, but by far cheaper and simpler to implement. Also in areas where people object to the presents of cameras, the option of using PIR sensors is an alternative.

3.5.10 Gas-Sensors and Electronic Noses

Many types of gas sensors are available to measure gas concentration in the air. Also available are gas sensors that are designed for specific tasks, e.g. detecting food or alcohol. Generally these sensors need to be pre-heated before a measurement can be taken, typically the heating time is around a minute [Figaro,02]. This results in a rather long delay before useful values are available; for many applications this is a severe constraint. In environment based sensing scenarios it is an option to always heat the sensors, for mobile devices the high energy consumption often rules out this option. The heating often consumes around one watt, for about one minute. The output is most often analog and can be interfaced to a MCU after amplification or directly.

These sensors are not applicable for general mobile context-aware applications due to the this high energy cost. Specific mobile appliances for security forces, fire fighters, and mining personal can however benefit from these sensors. These sensors can also provide interesting information for systems where sensors are embedded in the environment and power is a minor concern.

The term electronic nose describes multi-gas sensors or arrays of gas sensors that are usually used to recognise a particular smell or a variety of smells [Nose,02]. Technically they are more complex than sensors for a single gas. These sensors are usually developed for very specific applications, such as in the food industry.

3.5.11 Bio-Sensors

The term 'bio-sensors' is used to describe sensors that measure signals from life forms in various ways. Many bio-sensors are highly specialised and are only usable in lab conditions, whereas other sensors are fairly simple to integrate in personal devices. In general these sensors can be discriminated into two classes: invasive and non-invasive sensors. Invasive sensors are mainly used in medical appliances and are only applied when there is an evident medical indication to measure a certain parameter. For context-aware systems non-invasive sensors are of interest. In general their setup and application does not require a medical procedure. The following parameters can be measured with non-invasive sensors.

The pulse or heart rate indicates how calm, excited, or exhausted someone is. There are various sensors available to measure heart rate at different body points, e.g. on the finger, on the wrist, or on the chest. Including such a sensor in appliances that are designed to be used during exercising, means that this context information can be of great value.

Skin resistance is related to the ability of the skin to conduct an electrical current. Skin resistance gives an indication on the tension and excitement of the user, which is the same technique that is used in lie-detectors. However the resistance is also dependent on the type of skin (e.g. where on the body) and how the electrodes are fixed (e.g. just by holding, as finger ring, or with a plaster and conductive lubricant.

Muscle tension can be measured using electrodes that touch the skin. These sensors have been widely applied in bio-feedback systems. However muscle tension can be also used to recognise gestures and movements [Rekimoto,01].

Sensors are available that acquire blood pressure based on a module that observes blood flow. These sensors can be fairly simple, integrated in devices worn at the body. However the parameter of blood pressure is mainly interesting in medical applications. Non invasive sensors to measure the oxygen concentration in the blood are also available [Nonin,02].

Using electrodes placed on the body in a certain pattern the activation pulse for the heart can be measured, referred to as Electro Cardio Gram (ECG). By using electrodes

on the head of the user the activation of certain regions of the brain can be measured. The method known as Electron Encephala Gram (EEG) requires expensive equipment and qualified personnel to set it up.

In summary the value of these sensors is obvious for medical devices and bio-feedback applications. These sensors can also be deployed directly in appliances that are designed to be used during sports and exercise, and also for people working under extreme conditions.

Some of these sensors can be used to acquire information, or at least some hints about the emotional state of the user (e.g. how tensed he is, or how excited she is). These sensing technologies can also be used to create new input mechanisms [Affective,02], e.g. to be exploited in games [Moberg,02].

3.5.12 Zero-Power Sensors

When designing systems where the main concern is to save power sensors should be deployed to support that goal. In these settings sensor circuits are usually deployed to generate an external interrupt to wake up a MCU or a whole system that is in sleep mode. The main design goal is that the sensor system consumes only minimal power, or at optimum no power while nothing happens.

Typical sensors that can be used to design such wake-up mechanisms are motion switches (available to detect angle, shock, and vibration) and solar panels. The switches are attached to the device that they are open while the system is in sleep mode and that they close and generate a signal when the device is moved. The solar panel can be used together with a capacitor to generate a signal when the light level is changing.

3.6 Composition of Sensing Systems

When designing systems it is also an option to deploy multiple sensors to make a perception task easier or even feasible. Sensor fusion is investigated in automation and robotics for domains with a clearly defined sensing objective [Brooks,98].

In Ubiquitous Computing environments several options for creating a sensing infrastructure are possible. Table 4 illustrates the design space for deploying sensing

technologies. Sensors can either be in one place or they can be spatially distributed. When selecting sensors they can be homogeneous or heterogeneous. References to projects and publications are mentioned for each region. It is assumed that there is always communication between sensing modules.

A major issue, especially in a distributed networked setting, is where the sensors are placed with regard to the observer. For example having a sensor to can sense someone walking by can provide similar information, then a device that is worn and detects the movement of the wearer. This also makes it than necessary to consider to which identity a context or activity is assigned and how transformation between different observers can be realised. These decisions have also implications on privacy. Distribution aspects are assessed in more detail in chapter 6.

3.6.1 Sensor Arrays and Groups of Sensors

Putting a number of sensors of the same type into an array often provides additional information that is hard to get from just one sensor. To illustrate this consider the usage of microphone; using a single microphone it is quite hard (and needs extensive processing power) to separate different distributed sound sources. Having two microphones or an array of microphones physically distributed this task becomes rather simple and can be solved on a microcontroller as we have showed in [Schmidt,99b]. In our experiments we could see that it is often feasible to ease the perception task significantly by adding additional sensors of the same type to the system.

When building sensing technology into mobile and handheld devices it is often necessary to use multiple sensors of the same type placed at different physical positions of a device. Especially when there are little restrictions in the way a device is held or carried; this can help to ensure that at least one sensor gets a reading that is

	All sensors at same position	*Sensors distributed*
Homogeneous sensing system (one type of sensor)	e.g. orientation aware PDA, [Schmidt,98]	e.g. load sensing system, [Schmidt,02]
Heterogeneous sensing system (different types of sensors)	e.g. context-aware mobile phone, [Schmidt,99c]	e.g. distributed sensing boards, [SMART,02]

Table 4: Design space for deploying multiple sensors.

useful. A typical example is a light sensor: having only one on a handheld device may lead to the case that the user puts their finger on top of the sensor resulting in a false reading (as many people have experienced with cameras where often the light sensor is mounted in a way that it can be conveniently covered when holding the camera). Having multiple sensors, such situations can be detected or even avoided.

Grouping sensors of different types closely together, as we investigated in TEA [TEA,98] and in [Beadle,97], has the advantage that these devices are fairly easy to produce. The drawback however is that these sensors are for most applications not optimally placed.

3.6.2 Placement of Sensors

Where sensors are mounted or attached is of major importance and has a great influence on the quality of the data that is gathered. When putting all sensors together in one place, e.g. mounting them on a sensor badge, perception of contexts becomes much harder than for a case where sensors are physically distributed and placed at an optimal point. This is especially true when considering applications in wearable computing, where it is of great value to distribute sensors over the body to positions where the parameters of interest can be most easily read, [Laerhoven,00] and [Laerhoven,02].

Finding and selecting the 'right' position for a sensor is very much dependent on the contexts that should be recognised. E.g. when it is of interest what manual task the user is carrying out, mounting accelerometers on the wrist is quite useful. Whereas when contexts regarding movements, such walking or running, are of interest, then the hip is a good position to mount the accelerometer.

In the case of environment based sensors analysing the physical conditions of the space can provide important information. E.g. building a sensor that can tell that a room is occupied can be either realised by sensing the actual room or by sensing the entries to that room.

In most case, where there is no prior knowledge about sensor placement, the best position can be found by acquiring test data with sensors mounted at different positions, and then selecting the optimal position based on this data.

3.7 Perception Methods for Systems with Limited Resources

In this section, a selection of perception methods is discussed. The focus is on simple techniques that can be used on devices that have very limited processing power and memory. For some methods the approach was to approximate calculations usually carried out on powerful systems (with no real limitation regarding processing power and memory) in the best possible way on the limited system.

3.7.1 Basic Statistical Functions

The **average** of the data samples provided by a single sensor over a given time-window can be calculated with minimal cost[3]. Calculating the average is meaningful for data from nearly any sensor, e.g. light, acceleration, temperature, and pressure sensor.

The **median** is a valuable measure to eliminate extreme values and false readings in the signal that is supposed to be stable or slow changing. However if the sample size is very small the median may proof of little value. To avoid the need for a division in the calculation an uneven number of samples can be taken.

Calculating the **standard deviation** can give an indication on how stable a signal is, or how much change there is in the signal. The measure is less useful if it is known that the signal sometimes carries wrong values, because even a single value can distort the result.

The **range** of the samples collected can be easily calculated by finding the **minimum** and **maximum**. This can be done on the fly without the need to save all samples, by always updating minimum and maximum after each reading. Range is however very vulnerable to single false readings. These errors are avoided by the use of percentiles, e.g. using the **interquartile range** is more robust. Sorting the data and calculating the distance between values at one quarter and three-quarter is obviously more reliable than using the range, e.g. a few faulty values do not wreck the calculated feature. A compromise between both measures that can be calculated without storing all samples

[3] By selecting the number of samples over which the average is calculated as 2^n the division operation can be replaced by a shift. When calculating the average of 256 Byte values and storing the sum in a 2 Byte variable, instead of dividing only the MSB can be taken as result.

is to hold, not just minimum and maximum, but also the runner-up for minimum and maximum (or even the *n* smallest and *n* largest values).

An **indication** of the amount **of change** in a signal can be gained by calculating the sum of absolute differences between the average, and each data sample in a window. Summing up the differences between following samples can be done on the fly and gives information on how rapidly a signal changes.

3.7.2 Time Domain Analysis

To avoid the transformation into the frequency domain feature extraction procedures in the time domain can be used. This has been used in particular on data from accelerometers, light sensors and audio.

Finding the average value is computationally cheap and can be done on the fly with little need for memory. For audio the average itself has no meaning but it is useful to calculate further features. Knowing the average means that calculations on how often the average is crossed in a certain time and also the average distance between crossing the average, can be performed. It is also possible to calculate the distribution of the distances between crossing the average. This is an indicator for the base frequency and the stability of the base frequency in the signal. Counting the direction changes in the signal is also possible on the fly. The ratio between the average crossings and the direction changes gives an indication on the type of signal and allows discrimination between contexts. For example in the audio signal it is possible to discriminate music, speech, and noise, and in the acceleration signal it is possible to find characteristic values for certain patterns of movement. More details and an example are shown in Appendix A.1: Time Domain Analysis.

For fast changing signals like audio signals, the peaks or energy (root mean square) of the signal in small time windows (e.g. getting a indication every 100ms) provides information about the sampled data. Certain audio events (speaking of a word, ringing of the phone, applause, music) result in a characteristic series of values. See Appendix A.1: Time Domain Analysis.

3.7.3 Derivatives

For many signals it is of interest to find information about the change rather than about the absolute values. Calculation or estimation of the **first derivative** of the sensor data indicates the direction of change in the signal. This information is especially helpful to find transitions in the observed conditions, e.g. going into the dark or speeding up. In the simplest cases this can be estimated by checking whether or not the samples are continuously falling or rising. Another indication is to sum up the differences of consecutive samples.

Analysing **higher derivatives** provides information on how the changes that occur, are changing over time. A simple way of doing this is to hold a history of features (e.g. the changes in light) and analysing these features similarly.

3.7.4 Neural Networks

To provide abstract or symbolic information, it is necessary to process the calculated features and cues further. Neural networks can be set up so that they take the cues and features calculated as input and provide context or the context class as output. Many neural networks are computationally demanding. The following approach however can be implemented on very restricted hardware platforms.

Logical Neural Networks as described in [Aleksander,95] offer a computationally cheap method to learn and recognise patterns. In the learning phase the applied input patterns are transformed into binary vectors, which are then subdivided into shorter parts. For each class of input pattern a logical storage unit is used. The short patterns are then use as memory addresses of the assigned storage unit (e.g. internal or external RAM). For each pattern seen the value of the storage is 1. In the recalling phase an incoming pattern is also transformed into a binary vector and subdivided into parts. The output is then the class of the memory unit that has the most sub-patterns in common with the incoming vector. For an example see [Schmidt,96,p24ff]. Implementing learning as well as recall is feasible on very simple hardware. Depending on the number of contexts to be recognised and the size of the input vector additional storage is required.

It is also possible to implement **Backpropagation Neural Networks** on restricted hardware. Basically it is possible to implement small backpropagation networks directly on these devices, however due to storage restrictions and also to increase training speed, a distributed implementation is often preferable. This is a two step process. In the first phase, when the contexts are learned, the input vectors consisting of cues or sensors values, or a mixture of both, are acquired on the microcontroller device and communicated to a backend system (e.g. over serial line or RF to a PC). These data samples are annotated with the context they are recorded in. In the backend these data is used to train a backproagation network of appropriate size and structure. When the training process is finished and all weights are calculated these can then be coded into the recognition software that will run on the microcontroller device. By coding the weights directly into the recognition code the size of the network is mainly limited by the program memory.

Nearest neighbour matching is a very simple technique for pattern matching. When implementing this technique on a microcontroller it can be done with varying complexity depending on the requirements. In a simple implementation for each class a representative vector is calculated and stored during the learning phase. When the system is in operational mode, an incoming vector is compared to the stored sample vectors and the distance is calculated. The nearest neighbour is then selected as the class to which the input belongs. In the TEA project we used this approach in one of the experiments to recognise different motion patterns.

If the clusters are not known in advance using the Kohonen Self-Organizing Map (SOM) or one its many variants is another option [Fausett,94,p169ff]. The clustering algorithm is able to learn new clusters at any time and can also handle noisy data. To make the output meaningful, the produced clusters must be labelled with context names [Laerhoven,99]. The topology preserving property of the SOM makes it very probable that the nearest label will indeed be the right context. Generally, the longer the system is trained, the better the recognition becomes.

3.7.5 Rule Based Systems

The straight forward approach is to integrate rules while programming. This usually is done without much thought, when the domain is limited and easy to understand. This

is particularly simple when sensors map well to contexts of interest and the number of contexts is small. An example, developed in the project TEA, is a device which can detect the context ‘in a pocket’, ‘on the table’, ‘in the user’s hand’ by sensing acceleration, light, temperature and touch, for a similar experiment see Appendix A.2: A Simplified Rule Set.

To build rule based systems a two step process is used. In a first phase data is collected and transferred to a backend system. The data annotated with the contexts that the samples belong to, and then analysed to find appropriate rules. Knowing the rules resulting from the data analyses, these can then be implemented on the microcontroller device. This can be done by hard-coding the rules into the source code of the software that will run on the microcontroller. Another option is to build an interpreter that runs on the microcontroller and can take rules as input and interpret these.

A further option for systems where rules are applicable, but the borders between states are less clear, is the use of Fuzzy Logic [Zadeh,73], [Traeger,94].

3.8 A Perception Architecture for Context-Aware Systems

As described above there is a variety of sensors, low level perception methods, and high level abstraction algorithms available and usable, to acquire context. The following architecture offers a flexible and yet efficient framework to build perception systems and offers an abstraction for context-aware applications. The approach is to deploy a layered architecture, as depicted in Figure 3. The architecture consists of four layers, sensors, cues, contexts, and an application layer. With interfaces between the layers the architecture also caters for settings where sensors, cue extraction, context processing and applications are distributed, as detailed in chapter 6. Optionally the layers can be connected via a network.

3.8.1 Sensor Layer

On an architectural level sensors are components that can provide information about the world. No discrimination is made between physical sensors and logical sensors. Physical sensors are hardware components that measure parameters in the environment and provide the information on electronic level, typically as analog

output or as digital signals. Logical sensors are components that provide information that is not directly taken from the environment but represents information about the real world, e.g. a clock as a sensor that offers time and a server offering the current exchange rate. These sensors supply the information most often as digital signal over a common interface such as a serial data connection or an HTTP-connection.

More formally each sensor S_i is regarded as a time dependent function that returns a scalar, vector, or a symbolic value (X). A set (finite or infinite) of possible values (domain D) for each sensor is defined. This is a common way of describing sensors [Brooks,98]

$$S_i: t \rightarrow X_i$$

t *is the time (discrete),* $X_i \in D_i$, i *is the identification of the sensor*

To simplify calculations and regarding the fact that processing is done on digital systems the time is considered as a discrete variable.

To achieve exchangeability and hence flexibility for each sensor, a physical and logical interface is defined. The physical interface consists of the mechanical specification of the connector and the electrical specification. In Table 5 an example of these descriptions for three different sensors is given. For high level sensors the

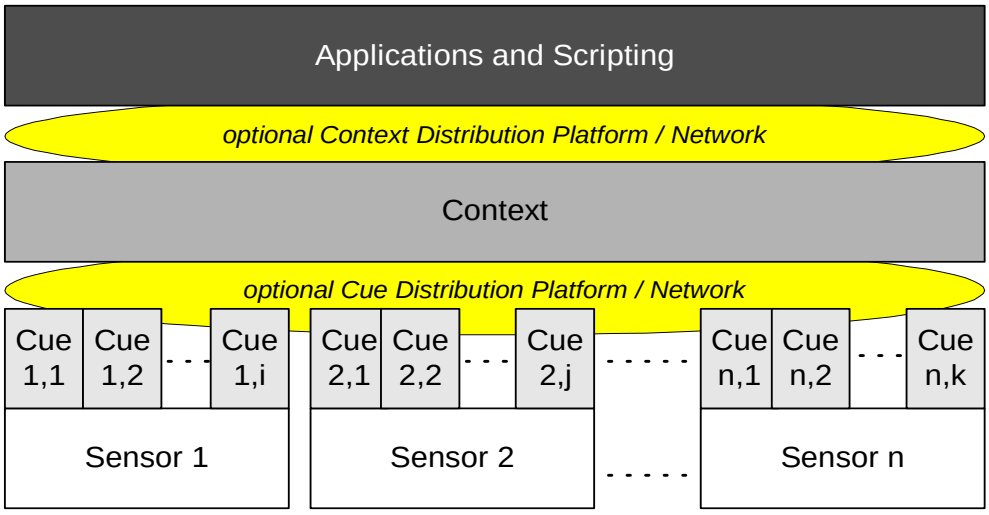


Figure 3: Layered Perception Architecture.

	GPS-Location sensor (e.g. Garmin Etrex Vista)	*Light sensor based on a light to voltage converter (e.g. TSL250)*	*Service on the internet that provides foreign exchange rate*
Mechanical specification	Female Sub-D connector 9 Pins, RS232 pinout	three pin hole-through soldering connection with 2.54mm grid where pin 1 is GND, pin 2 is Vdd, pin 3 is Vout, Vout is connected to an analog input of the MCU.	Ethernet connection
Electrical specification	+12/-12 Volt, RS232 specification	GND is 0V, Vdd is 5V and Vout is between 0 and Vdd,	Ethernet specification
Logical interface	NMEA format,	0V is dark, increasing voltage, in full light ouput is Vdd,	TCP / HTTP protocol spec. URL to get the exchange rate, data type of the answer.
Software access	Serial line API	Analog/digital conversion API	TCP/HTTP API

Table 5: Sensor Specification Examples.

level of detail for the specification is straightforward (e.g. as for a GPS receiver), whereas for low level electronic components the specification could be extended to include all the information available about a sensor. However if the specification is too detailed it fits only one sensor. To make it feasible to exchange and replace sensors with compatible ones the specification should only denote the really essential parameters for its operation.

The specification of a sensor always reaches a level where the output data can be directly accessed from the software running on the systems, e.g. on the microcontroller. The method of access is usually done over an interface, such as accessing an analog-to-digital converter or reading from the serial line.

3.8.2 Cue Layer

The concept of Cues provides an abstraction of physical and logical sensors. As seen from Figure 3, each cue is dependent on one single sensor, but using the data of one sensor, multiple cues can be calculated. In contrast the term feature is used, a general concept of abstraction where the input data does not necessarily originate from the same sensor.

A cue C_{ij} is regarded as a function taking the values of a single sensor *i* up to a certain time *t* as input and providing a symbolic or sub-symbolic output *Y*. A set (finite or infinite) of possible values (domain E) for each cue is defined.

$$C_j: S_i(t) \; x \; S_i(t-1) \; x \; \ldots \; x \; S_i(t-n) \rightarrow Y_{ij}$$

t *is the time (discrete),* $Y_{ij} \in E_{ij}$, $t \geq 0$ $n \geq 0$, *j is the identifier for the cue*

As seen from the definition, each cue is dependent on one single sensor but using the data of one sensor, multiple cues can be calculated. Cues are one way to reduce the amount of the data provided by the sensors. For physical sensors, introducing a layer for cues also can ease the calibration problem.

To implement cues all perception methods introduced earlier applied to the data from a single sensor can be used. In particular basic statistical functions and time domain analysis proved to be of great value. They can either provide a summary of the values over time or they help to extract features from the raw data that characterise the data over the last period of time.

The decision of what time window to use as basis for calculating the features, has implication on the meaning of the feature as well as for the implementation. In general longer time windows need more storage and the features have a longer delay before they are available. But longer time windows allow more analysis over time and hence often offer valuable information. When generating multiple cues that need data collected over a longer time window a buffer can be shared between all these cues calculated from one sensor [Chen,99]. The buffer is then a layer between the sensor and the cues. A reasonable time window depends on the sensor and ranges between 100ms and a few minutes.

When designing cues the properties of the sensor should be taken into account. Working on relative values or on changes in the signal is often useful to help eliminate effects introduced by de-calibration of sensors.

In building prototypes, the concept of cues proved to be very useful to make changes of the hardware transparent for the context recognition layer. When including new

sensors with different characteristics, only changes in the corresponding cues must be adapted.

3.8.3 Context Layer

As defined earlier a context is a description of the current situation on an abstract level that can be matched against previously specified situations. These descriptions consisting of conditions may or may not be human readable; in general they can be regarded as a function that takes cues as input and evaluates to true or false stating whether the current situation belongs to a context or not. The context layer holds all possible context descriptions that are relevant and modelled for a given scenario or a particular application. In a certain situation none, one, or more of the contexts may satisfy the relation that the situation belongs to the context. The evaluation of conditions to true or false can be enhanced by assigning a certainty to the decisions on the relationship.

Depending on the dimensionality of the data available from the output of the cue-layer different perception algorithms can be deployed to evaluate contexts. To realise a system working with just a few well understood cues and a small set of contexts, simple rules (e.g. first-order logic) can be used to determine whether or not the current situation belongs to a context.

The hardware constraints (e.g. memory, processing power) and the anticipated behaviour of the recognition algorithm (e.g. fixed, able to learn, and reaction time) also have to be taken into account when deciding on which method to use.

3.8.3.1 Learning

How to incorporate learning capabilities and learning behaviour into context-aware systems is a major design decision, see Table 6. Basically three concepts can be discriminated:

- No learning after the development is completed
- Dedicated learning phase during use
- Continuous learning

Concept of Learning/adaptation	*Usage*	*Algorithms*	*examples*
No learning, fixed	Contexts are globally valid	Design time data analysis	Static Rule based systems, Preset Supervised NN
Learning phase	Contexts are stable but different depending on the use case	Training and/or data analysis capabilities built in	Dynamic Rule based systems Supervised Neural nets
Fully adaptive, always learning	Contexts are changing over time	Adaptive algorithms	SOM, ISL [Hagras,02]

Table 6: Learning and adaptation.

When the contexts that should be recognised and the conditions that indicate these contexts are independent of where the system is applied, then no learning is required after the system has been developed and deployed. Based on collected data algorithms can be found beforehand that recognise these contexts. The conditions can be hard-coded; typically this can be done using rule based algorithms and supervised neural networks.

Learning capabilities can be incorporate into the system. In a specific learning phase knowledge is acquired. When contexts differ depending on the usage of a device, but are stable over time, a system using a dedicated learning process is required. An example is a device that offers different behaviour in the contexts ‘at home’ and ‘at work’. After purchasing a device it has to learn what the contexts are like in this specific case. This requires that the learning algorithms are implemented on the actual device. In this phase the conditions for a context are determined.

Having continuous learning built into a system is an option for devices that operate on contexts that may (slightly) change over time but still have the same meaning. To implement this, flexible algorithms are required and also mechanisms that supply, at least from time to time, the labels for recognised contexts. Algorithms such as the self-organising map can be used to realise this form of adaptive behaviour.

In chapter 6 this architecture is extended with capabilities for the distribution of sensing, processing and context usage.

3.9 Discussion

Having the right sensing infrastructure is just a prerequisite, to make use of it, perception and the ability to match the stimuli acquired with patterns that indicate actions is also an essential part. Senses, as we know them from biological systems, incorporate all these steps, whereas sensors in a technical context only provide very basic information.

When considering sensing and perception in Ubiquitous Computing it becomes inevitable to assess the cost introduced vs. the gain provided. In nearly all cases sensing is not for free, typical costs introduced are: higher power consumption, more fragile devices, and more complex hardware and software. For many applications sensing can provide new qualities that would otherwise not be possible.

Considering the appropriate sensing infrastructure to get a particular context is tricky, often it is easily feasible to detect a context when there are no restrictions (e.g. large number of sensors and unlimited processing power). In real settings the central issue is to create a minimal sensing and perception system that can provide the information required.

When using sensors to spot contexts and to detect situations, one has to be aware that sensing systems have limitations. As contexts often do not have well defined boundaries there will always be cases where it is not clear whether or not a situation belongs to a context or not. This is not necessarily a problem of a poor perception system; this problem may even apply when having several human observers who have to make the decision. Using a human observer as a simulation for the optimal context sensing system is motivated by this observation and discussed in more detail in chapter 8.

A further problem that can lead to ambiguities is that when creating systems, only a limited number of possible situations that should be matched by a context are assessed. Usually these are the situations that are expected by the designer. In real use however it is likely that situations no one considered before,will appear and possibly result in a context that was not desired. If the system is well engineered these exceptions are rare but still it will be not possible to guarantee 100% reliability.

In contrast to standard automation systems and also to applications in robotics sensing in Ubiquitous Computing has to consider **the human in the loop**. In process automation the tasks and requirements are usually well understood. Also the question as to what has to be sensed and how the sensor reading should be interpreted is clearly defined by the process. A process that is set up usually stays the same with all operational functions that were anticipated at design time. In robotics and in particular for autonomous robots sensing is also clearly related to a task, such as finding a path or trying to avoid hitting an obstacle. In Ubiquitous Computing where sensing is deployed, providing the system with a world model that relates to the user's world model, the sensing and perception task is less focused and often more general.

3.10 Summary

In the beginning of this chapter the concept of sensing and context acquisition is motivated by biology and in particular by looking at the senses that are around in nature. A terminology for situation, context, and their relationship is introduced. Context is considered as an abstract description of a type of situation. For a particular situation it can be determined whether or not this situation belongs to a context. The overall hypothesis that the sensory stimuli received in situations of the same type, at least in their discriminating features, are similar is stated.

From a more technical perspective the requirements on sensing with a specific focus on Ubiquitous Computing are analysed. Based on these requirements and drawn from the experience gathered in various projects, an overview on sensors and sensing systems that are interesting for context-aware computing is outlined. This is complemented by the consideration how the deployment of multiple sensors can ease perception tasks. A selection of perception methods with respect to systems with very limited resources is assessed. The sensors and algorithms discussed here are the foundation for the building blocks and libraries introduced in chapter 5.

In the final part of the chapter a layered perception architecture is introduced. Discriminating between sensors, cues, and contexts the architecture provides a framework for context acquisition systems for Ubiquitous Computing. This architecture is the groundwork for device and software frameworks introduced later.

Chapter 4

Modelling and Prototyping

Sensing and perception provides means to acquire context. In this chapter it will be investigated how context can be modelled and in particular how situation and context relate to entities. This results in a conceptual model, suggesting context as entity-based information. The corresponding implementation model is based on to the approach of embedding context and context-awareness into artefacts.

4.1 Context and Entities

The term context-awareness is used in various ways. However most often it is not explicitly clear how *context* is anchored in its environment, and who or what has *awareness* of this context. Usually it is assumed that this is implicitly clear.

This can be illustrated considering the example of a context-aware information system that is mainly based on location; for examples see [Brown,96], [Abowd,97], and [Davies,98]. The context 'location' does not have meaning on its own it is always related to a user, device, or application. Awareness is either performed by the device, by the application, or by the user. Similarly the models developed for context aware applications are often opportunistic and strongly related to the technologies deployed. In our understanding the following properties of context are central:

- Each context is anchored in an entity.
- Context-awareness is always related to an entity.

An entity is a place, an artefact, a subject, a device, an application, another context, or a group of these. When creating sensing systems that supply context about an entity the domain knowledge that is available on the entity can be exploited, leading to the following hypothesis:

> **Hypothesis 2: The domain knowledge about a specific single entity is more universal and easier to establish than the domain knowledge of a complex system, and hence it is simpler to identify and implement contexts on entity level than on system level.**

To illustrate the concept consider the example of a coffee cup with the contexts described in Table 7. It can be observed that these contexts of single artefacts are greatly independent on the general situation of use, so on this level there is no difference whether the cup is used at home, in the office, or in a restaurant.

The approach followed here is to take these issues into account when designing context-aware systems. The following definition underlines the basic relationship between context and entity.

> Definition: **Context-Aware Entity**
>
> A context-aware entity is an entity that has contexts that are anchored in it.

In a context-aware information system a context-aware entity can provide and/or use context. Modelling context around entities seems to be a natural way of building such systems, because on the level of objects the concept of contexts and their relation to

Type of entity	*Entity*	*Examples of typical Contexts*	*Contexts relations in example*
Person	Athlete	Running, walking, sitting, cycling, lying.	Exclusive
Body part	Hand	Moving, moving fast, still.	Non-exclusive
Artefact	Coffee cup	Empty, full, hot, cold, moved, drank from.	Non-exclusive
Part of an artefact	Handle of a coffee cup	held, not-held	Exclusive

Table 7: Examples of entities with typical contexts assigned.

situations is in most cases very well understood. The shared understanding of situations and contexts relating to artefacts, as communicated in every day language, offers a starting point with little complexity. Relating situations and tasks to objects and properties is a bottom-up approach to understanding and modelling context. When connecting contexts to artefacts an inherently decentralised model of the world is anticipated, because artefacts are distributed in the real world.

For many scenarios in Ubiquitous Computing it is a prerequisite to have access to a digital representation of the real world. In the bottom-up approach introduced here the main goal when constructing aware artefacts is to create a digital shadow of the real world, similar to the goal defined in the sentient computing project [Hopper,99].

This is a more general approach than providing a specific context for a specific application. Having a complete representation of the world would require that all artefacts in the real world are completely aware of everything around them and it seems at this point in time a distant goal. However, as our research indicates having some artefacts that are aware to some degree already provides a system with an impression about the surrounding world. Furthermore, the assumption is that the digital shadow gets clearer with each aware artefact added to the scene and also as the awareness of single artefacts gets better.

This *bottom-up context-awareness* is an approach to model, design and implement context-aware systems. Here basically for each artefact the main contexts are modelled that are directly related to this artefact, accepting that this may be incomplete, rather than trying to model the whole world from a top-down perspective.

Assuming that many entities are aware of their contexts it is appealing that there can be more context information by combining the context knowledge that is around to get a better understanding of what happens in the real-world. In particular two ways to discriminate are:

- Creating context for entities where the contexts of their sub-entities are known
- Establishing context for a group of entities that have a relationship

In the first case this is a compositional approach, where the *is-part-of* relationship is exploited. Whereas the second one is more general; examples of such relationships are spatial, temporal and thematic arrangements of entities. The contexts can be in both cases either very basic to contexts of the entity or groups of entities or contexts that are semantically on a more abstract level. This observation leads to the following hypothesis:

> **Hypothesis 3: Contexts for an entity or a group of entities can be established by fusing the contexts of entities that make up the entity or the group. Thereby artefact centric context enables versatile uses and becomes the foundation for a platform for applications.**

The strong relationship between context and entities motivates the conceptual model. The following major issues are considered in the creation of the conceptual context model and are also significant for the implementation model:

- Conceptual Bond between Context and Entity
  Context and entities are in many cases tied together. This conceptual bond is relevant to context acquisition and also to context use. Context can often be described by describing entities and their handling. Sensing on entity level carries additional information related to the domain knowledge of the entity.

- Context on Entity Level
  Modelling and implementing context acquisition and use on artefact level reduces the complexity. Sub-dividing entities into smaller entities modelling becomes a structured process. Similarly implementing context acquisition and context use for smaller units is in general simpler.

- Composing Context, Extensibility
  Providing context in a natural structure, related to entities, a potential for further compositions and extensions is introduced. Similar to approaches in Object Oriented modelling it is assumed that by these means the versatility and reusability of components is increased.

- Roles of Artefacts

  Entities can act differently within a system. Basically they can provide context, they can use context, and they can do both. Later entities will be referred to as context suppliers and context consumers to mark their roles.

4.2 A Conceptual Model: Bottom-up Context

When modelling context in a bottom-up approach it is always related to an entity. The entity is a subject, a part of a subject, an artefact, or a part of an artefact. Contexts related to subjects, as central to wearable computing, are concerned with a person or with parts of the body of a person. Context relates to artefacts and this is a key aspect to gain a digital shadow of the real world for use in Ubiquitous Computing. In Table 7 examples of contexts related to entities are shown. The table also illustrates two other issues:

- There is an inherent relationship between entities, especially when one entity is a part of another.

- The list of contexts that can be related to one entity is never complete and it is subjective.

When modelling bottom-up contexts can informally be seen as *what can be observed about an entity*. Typically contexts represent the state of an entity or the relationship of an entity to other entities. The latter one also includes the way artefacts are used by humans, as they are an important form of context. One assumption on the relationships between entities is that if an entity A is a part of entity B then it is useful to know the contexts of A to determine the contexts of B. In reverse it is also assumed that the context of an entity can be estimated by knowing the contexts of all its subparts.

When designing aware artefacts it becomes a major issue: how to select the contexts that should be included for a specific entity? Especially while knowing that the list of contexts for an entity is unlikely to be comprehensive, the decision when to stop modelling is non-trivial. The suggested way to find out what contexts are relevant for an entity is to monitor typical situations which the entity is in. For example asking questions such as ‘what is the entity doing?’ or ‘what is it used for?’. Knowing the

possible situations in which an entity is involved the most common ones are selected and represented as contexts. At this point decisions have to be made weighting the gain vs. cost; and this can in the real world hardly be done without looking at possible applications. To minimise the cost odd situations or situations that appear to be unimportant are left out. This includes the risk of missing contexts that may be quite important on a higher level; however it is practically impossible to include all situations.

The bottom-up approach suggested here, focuses on contexts that are important on the entity level rather than considering high levels of abstraction at this point. This assumes that if each single entity is modelled with its most important contexts it is quite likely that the information needed on a higher level can also be deduced from this information. Going this way, the resulting system is more general, and it is more likely that applications that emerge can be supported even after certain artefacts have been made context-aware without anticipating them. However, it has always to be remembered that selecting contexts for an entity is a deliberate choice of the designer.

Contexts that are selected for an entity can have different relationships between each other. In simple cases they are exclusive, meaning that only one context can be valid in a given situation. The contexts for the ‘handle of the coffee cup’ as described in Table 7 are an example for exclusive contexts. Whereas the contexts stated for the ‘coffee cup’ are non-exclusive. So it can be the case that there is a situation so that the contexts ‘cup full’ and ‘cup is hot’ are both valid. The design decision whether or not to allow non-exclusive contexts has implications for building the recognition system as well as for the programming model when using context. In general recognition is easier to realise when contexts are exclusive. In the programming model it has to be considered that contexts can appear at the same time and are valid over an interval in time, which is quite different from standard event models used in GUIs and suggested in [Salber,99].

4.3 An Implementation Model: Context Aware Artefacts

Analogous to the conceptual model the implementation model is entity centric. Concentrating on contexts that are related to an entity there are various ways in which the context can be acquired, as introduced in chapter 3. In general, two different

sensing approaches can be discriminated: intrinsic and extrinsic sensing. In the case of the first one sensing is built-in to the artefact and in the second case the entity is observed. In Table 8 examples are given. In many cases there is also a combination of both approaches used to realise systems. In the case where only the artefact itself uses the context this can be realised with intrinsic sensing without communication to other entities, however obvious implication of extrinsic sensing in this case is the need for communication. In the case where the observer is using context it is the other way round. In the general case when context is made available to the system, communication is needed in both cases, see Table 9.

Communication is most often understood as connection between various artefacts where information can be exchanged instantly; sometimes this is referred to as online-communication. In specific cases, where context is collected and used asynchronously [Thede,01], communication can be realised only at specific synchronisation points. An example is a piece of garment that records the contexts of the wearer and put the time stamped contexts into a database when put a coat hanger.

In the process of modelling and designing context-aware systems using the bottom-up approach, several issues have to be regarded, most notably:

- **Context Selection**. A central design decision is to select the contexts that are regarded as relevant for an entity.

- **Subdivision**. By subdividing entities in sub-entities the problem of modelling and recognising contexts can also be sub-divided. An example is to decide whether to model a coffee cup as one object or to model its sub parts, e.g. the handle, the bottom, and the container.

- **Exclusive vs. non-exclusive Contexts**. When selecting contexts for a particular entity the decision of what contexts to include has to be made. When

Intrinsic sensing (built-in)	MediaCup, Context-aware Mobile Phone, Weight table
Extrinsic sensing (observer)	Video capture Environmental Sensors

Table 8: Examples of intrinsic and extrinsic sensing.

modelling the choice between using only contexts that exclude each other or contexts that are non-exclusive can be made.

- **Intrinsic vs. Extrinsic Sensing**. The selection of contexts and also other constraints inherent to objects and their usage have implications how a sensing system can be implemented. In particular the question has to be addressed whether sensing is built into an entity or sensing is provided by observation, or a combination of both. This has consequences for the communication requirements.

- **Synchronous vs. Asynchronous Context use**. A further design issue is the way the information about context is used in the system. Basically it can be discriminated between an immediate synchronous (if used in interfaces often with real-time constraints) or a time-stamped context logging with asynchronous use.

For most entities there is also domain knowledge of how far reaching the implications of context of this specific entity are. Therefore it is also feasible to determine how far a context of a specific artefact should be visible in a system. The domain knowledge of how far a context is visible may have implications and is not as general as the contexts themselves, because it also depends on the environment where an entity is in, however the knowledge available here is still more general than on a system level. A more precise estimation of an appropriate range of visibility of context can only be established by knowing the applications. The notion of spatial relevance is elaborated in chapter 6.

	Context user		
	Entity	*Observer*	*Anyone*
Intrinsic sensing by the entity	No communication	communication	communication
Extrinsic sensing by the artefact	Communication	No communication	communication
combined sensing	communication	communication	communication

Table 9: Communication requirements depending on sensing paradigm.

4.4 Prototyping Context Aware Artefacts

To build and use system prototypes to develop an understanding of the design issues in ubiquitous computing was fundamental to the pioneering work of Weiser at PARC [Weiser,91], [Want,95]. Since then this method has been adopted as principal research approach by many researchers in the ubiquitous computing community. As ubiquitous computing engages with new device concepts, with new interactions between devices and their physical environment, and with embedding of devices in everyday objects and structures, this commonly involves prototyping of physical system components alongside the development of communication, system and user interaction software. Physical prototyping however often involves tedious tasks such as circuit board design and selection of electronics components at a very low level.

When building prototypes it can be observed that learning occurs at different stage in the process most notably when:

- An idea is transferred into a prototype
- People are getting the prototype to work
- Prototypes are used to communicate ideas and inspire
- Prototypes are deployed in a living lab
- The prototypes are used in studies

From the understanding gained when building prototypes more general concepts such as models, patterns, and architectures evolve.

Within the research leading to this thesis a number of different prototypes have been developed, partly at TecO at the University of Karlsruhe (Germany) and partly at Lancaster University (UK). Most importantly the following:

- Different context-aware PDAs [Schmidt,98], [Schmidt,99], [Schmidt,00a],
- A prototype of a context-aware mobile phone [Schmidt,99a], [Schmidt,99c], [Schmidt,00],

- A mug equipped with perception technology [Beigl,01], [Gellersen,00], [Gellersen,02],
- Context-aware garments [Schmidt,99b], [Schmidt,00b], [Laerhoven,02],
- Load sensing floor and furniture [Schmidt,02], [Schmidt,02a].

The development of these prototypes served multiple purposes. First it illustrates that implementing certain context aware artefacts is feasible in environments with constrained resources. Secondly they provide evidence for a proof of concept of the conceptual and implementation model of context aware artefacts, based on a bottom-up approach.

In this chapter the context-aware mobile phone and the load sensing environment will be described in more detail. The development and implementation of these prototypes exemplify the approach. In the latter section it is shown how to generalise from prototypes to generally applicable patterns of context aware artefacts.

The approach to prototyping was similar for most of the entities. First determine relevant contexts for the entity by observing how the artefact is used or deployed. Then the main constraints for the artefact are identified and used to define the requirements that the design had to conform with. Weighting the contexts of interest vs. the requirements and looking for a compromise results in an implementation that supplies contexts related to the artefact. These contexts are then used to make applications context-aware.

4.4.1 Context-aware Mobile Phone

In the project TEA ("Technology for Enabling Awareness") [TEA,98] we explored the possibilities for building add-on devices that supply context to a host system. In particular mobile phones were investigated as a potential host platform that can make use of context.

The project was carried out in two phases. In the first phase the basic feasibility was assessed with regard to the requirements and constrains imposed by the application scenario. In a second phase a prototype was developed and integrated into a

commercial mobile phone. Using this prototype the added value provided by context to applications was investigated, in particular the possibility of a device to adapt transparently to the current situation.

4.4.1.1 Phase 1: TEA Feasibility Study

Hardware, software, and applications were developed to build a demonstrator to prove the feasibility of making a context-aware mobile phone. In particular it was of major interest how contextual knowledge that is meaningful in the domain of a mobile phone, using low level sensors, can be gained. As the system was also built as an experimentation platform a main requirement for the prototype was flexibility. In particular the system should offer an efficient tool for finding the appropriate sensors and recognition algorithms.

The experimental setup used is depicted in Figure 4. The main components are a custom developed sensor board, a portable computer and a mobile phone. The sensor board and the mobile phone are connected over serial line to the host.

Sensor Board Hardware

To acquire information about the environment several sensors are used. They have been chosen to mimic typical human senses, as well as more subtle environmental

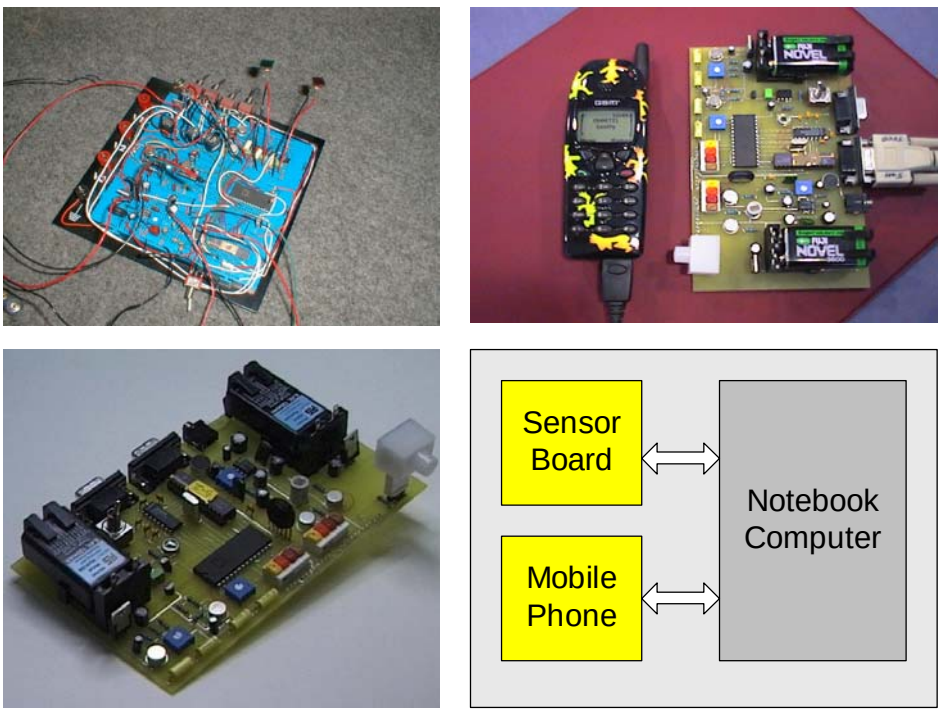


Figure 4: TEA hardware and system setup for the feasibility study.

parameters. An outline of the schematic is given in Figure 5. The photodiode yields both a nominal light level (as experienced by humans) and any oscillations from artificial sources (not a human sense). It is sampled at a rate of approximately once per millisecond, but only for a few hundred milliseconds at a time, allowing other signals to be multiplexed in. The two accelerometers provide tilt and vibration measurements in two axes; the signal is filtered down to 200 Hz. The passive IR sensor detects the proximity of humans or other heat-generating objects. This sensor is sampled at the same rate as the photodiodes. The temperature and pressure sensors each provide a conditioned signal between 0 and +5 volts directly, and need no amplification. These sensors are sampled a few times a second. Sampled at the same rate as the temperature and pressure sensors is a CO gas sensor. The MCU (PIC16C73) controls the heating and reading of this sensor. Each of the sensors provides an analog signal between 0 and 5 volts which is read by the 8 bit, 8 channel analog-to-digital converter.

The micro-controller oversees the timing of the analog-to-digital converter and the sensors as well as manipulating the data from the analog-to-digital converter's bus to the RS-232 serial line. Finally, the serial line connects to the data-gathering computer (Host). Higher bandwidth signals like the accelerometers and photodiodes are polled

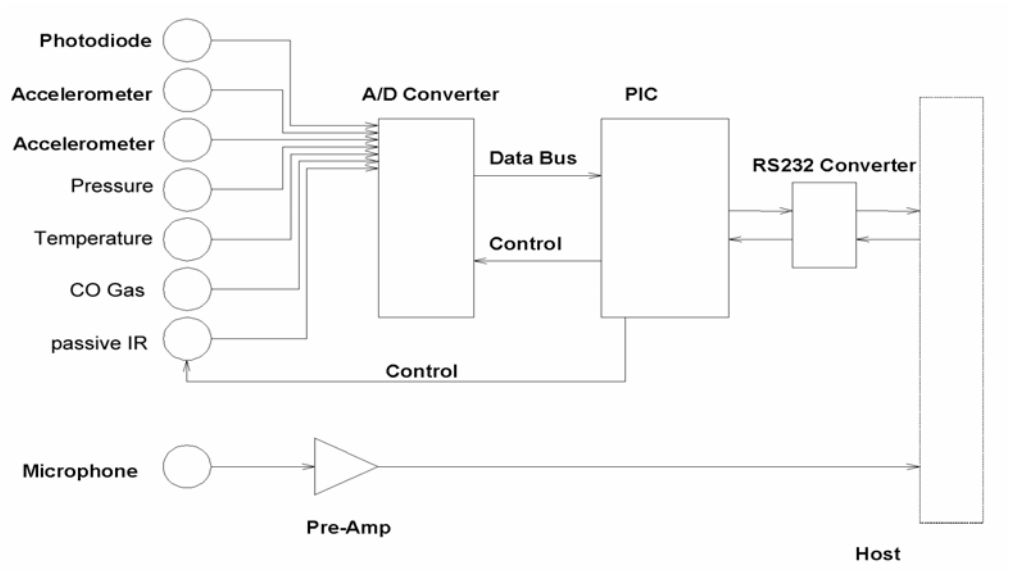


Figure 5: Schematic of the first generation sensor board.

often, on the order of every millisecond, while slower signals like temperature are only polled once a second

Another requirement of the system is mobility. In order to simulate mobile devices the board also has to meet certain size constraints. Here the compromise between experimentation flexibility and mobility resulted in a PCB size of 100mm x160 mm. See Figure 4 for the hardware implementation.

Off-line Data Acquisition

The sensor board sends periodically a block of data, representing the digitised sensor outputs, through its serial port. In the off-line data acquisition experiments the sensor board is connected to a notebook computer, which receives and stores the data. Using a specifically developed reader software, data is collected and annotated before written into a file. The annotations contain the time the sample was taken and also the manually assigned label describing the situation.

To gather data the sensing system is deployed in typical situations. For each situation, readings are taken over a certain time (between a few seconds and up to an hour). The experiments are repeated at various physical locations and with slightly different settings (but being considered the same situation). In particular the following data sets were collected.

- Set 1: holding device in hand vs. device in a suitcase vs. device on a table
- Set 2: walking while using the device vs. stationary usage
- Set 3: using the device inside a building vs. using the device outside
- Set 4: in car vs. on a bus vs. on a train.
- Set 5: having a device in a stationary car vs. in a moving car

The data sets are then analysed using data analysis and visualisation tools. Based on the raw data it is assessed what prediction systems and learning algorithms are most appropriate. A simple way to get an impression of the data, is by plotting the output of all sensors directly on a time scale in parallel.

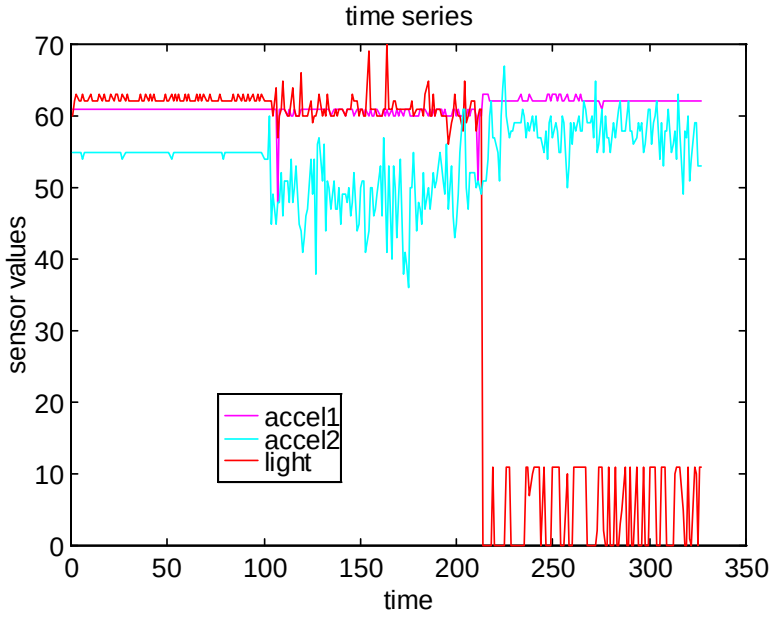


Figure 6: Example time series plot of sensor data.

The time series plot in Figure 6 shows the sensor values of the acceleration sensors and the light sensor in three different contexts. Initially, the sensor board was placed on a table and remained there for about 100 seconds. After this period, the device was taken along and stayed in the hands of its user for another 100 seconds. Finally, the TEA board was put in a suitcase for yet another 100 seconds. In the plot the context, or situations, can clearly be discriminated by a human observer.

Real-time Demonstrator

Based on the results from the data analysis methods have been developed and selected to implement a real-time recognition system. A set of basic statistical functions is used to calculate cues individually for each of the sensor values. To determine the contexts sets of logical rules are used. These rules are identified by analysing the data collected in different situations, as described in the last section. The complete recognition system is implemented on the notebook computer, calculating the contexts in real time from the sensor data acquired. The contexts recognised by the demonstrator are based on the data sets introduced in the last section. Each of the sets contains only exclusive contexts. Working with exclusive contexts makes the development of the recognition algorithm easier and also simplifies the development of applications. For details see Appendix A.2: A Simplified Rule Set.

To demonstrate the feasibility of context aware applications a mobile phone is manipulated using the contextual information. The context enhanced applications running on the Nokia 6110 mobile phone receives information about the current situation from the recognition software. The profiles of the mobile phone are selected automatically based on the recognised context.

The experiment shows that different contexts can be recognised using a subset of the simple sensors from the initially built board. Given the feasibility we went to the next phase: prototyping a context-aware mobile phone.

4.4.1.2 Phase 2: Prototyping a Context-Aware Phone

Based on the results of the feasibility study the design was revised. To implement a deployable prototype the portable notebook computer had to be eliminated from the system. Furthermore, the sensor board had to be integrated with the actual mobile phone to make it usable. The aim was to design a prototype of an integrated device in a way that it can be used as a normal phone with additional capabilities. Therefore size, weight, and also energy consumption became major requirements in the development.

Hardware

The redesign of the hardware resulted in a sensing and processing board that can fit, together with the phone battery into the case of the enlarged battery pack on the back of the phone. The selected sensors are placed at various positions in and on the phone body, to optimise the data that can be gathered.

An additional battery is used to power the sensor and processing board. The communication between the context-sensing board and the phone is realised using a wired serial line connection (RS232 at 3.3V). By these means the resulting prototype is still the size of a standard mobile phone. In particular the phone used (a Nokia 6110) looked as if the standard battery was replaced by a long life battery, see Figure 7.

The selection of sensors is based on the results and experience gathered in the first phase. In particular the pressure and gas sensor are eliminated from the design. The contribution of the pressure sensor was weighted against the size added to the board if

included. The power consumption of the gas sensor, in particular for heating the sensor, is too high and also the cues produced by the sensor are weak and therefore this sensor is also removed from the design. The PIR sensor is too large in size to be included into a phone but the information of proximity is rather valuable. To resolve this, the PIR is replaced by a touch sensor on the surface of the phone body. As the light sensor provides a major contribution two light sensors are included in the new design, distributed over the phone. The acceleration is of great values to determine contexts, especially user interaction, therefore in the revised design the two analog accelerometers are replaced with an integrated two axis digital accelerometer.

To make the board simpler, smaller and also more energy efficient the external A/D-converter is removed from the design. The MCU is exchange for a newer version, a PIC16F877 that offers 8 analog inputs and is also flash programmable. The analog sensors are directly connected to the analog inputs of the microcontroller. Also the fact that the MCU is flash-programmable eases the development process. For communication the built-in UART of the MCU is used and connected via serial connection to the phone. The wiring is done internally.

Figure 7: The sensor board and the enhanced mobile phone prototype.

Software

The context recognition software is realised completely on the MCU. The used MCU influenced the development significantly. The four major parts of the software, sensor reading, cue extraction, context calculation, and communication, had to be implemented on 8k of 14-bit words of program memory and using 368 Bytes of RAM.

The sensor reading and cue extraction is realised in a single loop to save RAM. For each sensor raw values are read into a buffer and then the features are calculated on this buffer. This is repeated for all sensors. The number of values and the speed of the reading are dependent on the type of sensor and in accordance with what was found out in the first phase. The cues that are calculated are statistical functions, mainly to reduce the amount of data that needs to be stored over one cycle before the context calculation.

Before implementing context recognition, example sensor readings for various contexts have been taken. The readings were taken using the sensor arrangement built into the phone. Cues are already processes on the board in phone. The output is used for further analyses providing a set of sample feature vectors for each context. Two recognition approaches are implemented.

One approach for context recognition is rule based and the rules are hard-wired in the program code. These rules are determined by analysing the sample vectors. Furthermore, common sense knowledge has been coded into the rules as well (e.g. between the context ‘on the table’ and ‘in a pocket’ there is probably a context ‘in the hand’).

The other approach is to use the sample vectors to calculate representative vectors for each of the contexts. Using these vectors as patterns a nearest neighbour matching algorithm is implemented. During run-time for a new feature vector the closest pattern is determined, as the winning context.

The communication to the phone is realised by using the propriety protocol of the phone. In particular the context-board mimicked a Nokia Data Suite running on a PC, which can be used to manipulate the settings of the phone.

On the phone no additional software is needed. To specify the behaviour of the phone for different contexts the appropriate ‘profiles’ are configured. When a context is recognised and communicated to the phone, the phone switches to the related profile and by these means adapts the behaviour that is specified for this context.

Demonstration and Evaluation

In the demonstration and test scenarios contexts that are similar to the contexts explored in phase one are used. In particular for validation and demonstration purpose the following contexts are implemented and successfully recognised: ‘Hand’, ‘Table’, ‘Box’, ‘Pocket’, ‘Outside’, and ‘General’. The behaviour of the phone is as follows:

- **Hand**. When the user holds the phone in their hand, the audio alarm is not used. The phone rings by vibrating.

- **Table**. Here a meeting situation is assumed. The phone is almost silent. Incoming call is indicated by a very gentle sound.

- **Box**. Phone is silent. Here it is assumed that the phone is put a way in a box or suitcase, and must be silent. The phone still receives calls, so that the callers' numbers can be recalled later.

- **Pocket**. Here the ring volume goes higher and vibra-alarm is on.

- **Outside**. Here the ring volume goes as high as possible and vibra-alarm is on. All possible ways to get the users attention are used.

- **General**. General mode is used when none of the above applies. The phone is at standard settings.

Changes in the behaviour of the phone for particular contexts can be applied by the user by changing the profiles. A different set of contexts however requires reprogramming of the context acquisition board.

4.4.2 Weight laboratory – Context-Aware Floor and Furniture

The weight laboratory was prototyped to investigate the utility of load sensing for context acquisition. In general three context primitives can be extracted from sensory

observations on load-sensitive surfaces: weight, position, and type of interaction. Weight is an obvious contextual parameter that can and has been used for instance as key to identify objects. Novel in this approach is to use load sensing for acquisition of object position and interaction events.

The research was carried out in two phases. First a series of experiments to demonstrate that high accuracy object positioning, and classification of interactions is feasible; this was accomplished. The experiments explicitly consider conditions in everyday environments, such as pre-loading of surfaces with a multitude of objects. In the second phase the experience and results from the feasibility study are used to design and implement a living laboratory environment where load sensing is built in.

4.4.2.1 Load Sensing Feasibility Study

Before building a context-acquisition system based on load sensing technology a number of issues had to be resolved. In particular it was important to understand the requirements and constraints imposed by such systems. Furthermore, it is also of great interest how accurate and robust such technologies can be deployed in everyday environments. To answer these questions a number of experiments were carried out.

Determining 2-D Position of Objects on Surfaces

For the experiment a table-top is placed on four industrial load cells each of which can detect forces of up to 500N. The load cells are placed at the corners of the table-top and each connected to a commercial signal conditioning unit. The conditioning units are in turn fed to a standard 16-bit Analog to Digital Converter (ADC) which links to the serial line of the PC. As these components are normally used for scales, the sampling frequency is necessarily rather low (each load cell can be read up to 4 times a second). The resolution of load sensing in this setup is approximately 16g.

To detect the position of an object the centre of pressure on the surface is calculated based on the load measured at each corner of the table. The overall force on the surface introduced by an object placed on the surface at (x,y) is denoted by F_x. The setup assumes static forces, so the sum of all 4 load cells F_1, F_2, F_3, and F_4 is equal to F_x, see equation 1 and Figure 8. If there is already an object (or the weight of the table-top itself, or multiple objects) on the table-top that is represented by $F0_x$ that can be measured by the forces on each load cell $F0_1$, $F0_2$, $F0_3$, and $F0_4$, see equation 2,

then these need to be incorporate into the algorithm for computing the position of a new object. The algorithm for identifying the position of a new object on a pre-loaded surface is described by equation 3 and 4 below.

$$F_x = F_1 + F_2 + F_3 + F_4 \qquad \textbf{(Equation 1)}$$

$$F0_x = F0_1 + F0_2 + F0_3 + F0_4 \qquad \textbf{(Equation 2)}$$

$$x = x_{\max} \frac{(F_2 - F0_2) + (F_3 - F0_3)}{(F_x - F0_x)} \qquad \textbf{(Equation 3)}$$

$$y = y_{\max} \frac{(F_3 - F0_3) + (F_4 - F0_4)}{(F_x - F0_x)} \qquad \textbf{(Equation 4)}$$

To carry out the experiments a Visual Basic program was developed, which reads periodically from the ADC, calculates the point of pressure, and visualises the result. In the program the user can manually reset the preload by pressing a button, storing the currently measured forces into $F0_1$, $F0_2$, $F0_3$, and $F0_4$.

In the first experiment, the position of objects on the table-top is detected. Before each object was placed on the surface, the preload $F0_1$, $F0_2$, $F0_3$, and $F0_4$ (resulting from the table top itself) is measured and stored. Then selected six well distributed positions are marked on the surface. The x and y coordinates of each position is measured manually and recorded for later comparison. Objects are systematically placed, one object at a time, onto the selected points. After the object is put down and the values stabilised, the load values F_1, F_2, F_3, and F_4, are measured and its position

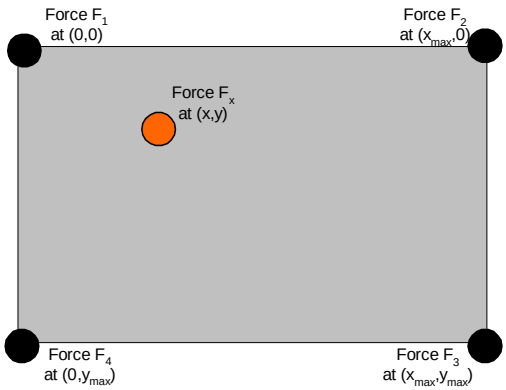


Figure 8: Forces on a surface used to determine the 2-D position of objects.

is calculated. This is repeated multiple times and for different objects (ranging between 100g and 5kg). The results of the experiment show that it is possible to achieve an accuracy of about 2% of the surface length in each dimension.

To simulate a more realistic environment, the experiment is repeated with a number of different objects already placed on the table (overall weight about 34kg). The table top is also covered with a tablecloth. Figure 9 illustrates this setup. The same set of measurement tests are repeated, whereby the objects are placed on top of the objects already on the table (on the TV-set, on the magazine, and the on the tablecloth).

The results we obtained are similar to those obtained in the experiment with the empty table. This experiment shows that using load sensing the static position of an object on a surface can be recognised irrespective of objects already on the surface or between the surface and the new object. The success of such an experiment also leads to postulate that the approach for object detection is deployable in non-lab environments and is especially well suited to Ubiquitous Computing settings.

Recognising Interaction on a Load Sensing Surface

As the world is not static and humans interact with objects, place them on surfaces and remove them again it is of interest to recognise events that relate to these actions. In the second series of experiments, it is explored how interaction resulting from events on a load sensitive surface can be recognised with a simple algorithmic approach.

Figure 9: The experimental setup; objects are stationary on a table while the position of an added object is detected.

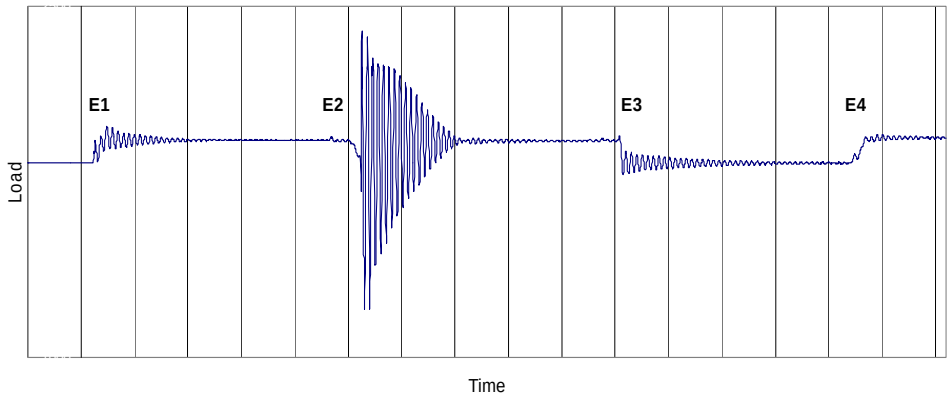


Figure 10: The graph shows the raw signals representing load change recorded over time. An object is placed on the surface at position E1 and E4. At E2 an object is knocked over and at E3 the object is removed from the surface.

The setup used for the experiment is a wooden table-top, 80x80cm, resting on 4 industrial load cells. Each of the load cells can handle a maximum load of 20N. The weight of the table-top is about 1kg. For this setup a specific hardware that allows a higher sampling rate (about 250Hz, as suggested in [Addlesee,97]) was designed. The microcontroller based system is connected to a PC via a serial line. The objects used to generate the events in this experiment are a 500ml water bottle (520g) and a book (~200g). More details are published in [Schmidt,02].

To detect the events an algorithm which considers the last 500ms using a sliding window and the sum of load (F_x) is used. At the selected sampling frequency, this equates to the last 125 sample values. At each sample interval the analysis algorithm is run, for details see Appendix A.3: Recognising Events on a Surface.

In the experiment it is investigated whether or not it is possible to recognise different events based on the features extracted from the data gathered. The focus is on the most important primitives: *putting objects down onto the surface* and *removing objects from the surface*. A further event: *knocking an object over* which is already on the surface is also included.

Overall, this experiment results in 70 recorded events for analysis. Using the simple algorithms 94% of the events were classified correctly, 6% were missed, and no events were misclassified. An example of a raw data stream is visualised in Figure 10.

The experiment was repeated on a table that was covered with a tablecloth. On the table were also 4 static objects, a notebook computer (about 2.2kg), a book (about 500g), a newspaper (about 200g) and a water bottle (about 520g). The same sets of interactions were performed. About half of the interactions were made on the tablecloth and the remaining half on top of static objects. Out of the 70 events recorded, the algorithm could classify 96% correctly, 4% were missed.

As seen from the results of the experiments, it is feasible to detect basic interaction events with a high degree of probability, even when using simple algorithms. It is also observable that covering the surface has a negligible effect on the data that was recorded. The result that preload has little influence on detecting the events or identifying an object's position, suggests that it is feasible to implement a system that can dynamically track objects that are added, moved, or removed from surfaces in everyday environments.

4.4.2.2 Prototyping a Weight Laboratory

To explore load sensing to acquire rich context unobtrusively in everyday environments a "weight lab" as an integral part of our living lab area has been designed and implemented. In the weight lab, a number of surfaces are load sensitive and equipped with networked data acquisition units. The setup was not driven by a single application, but developed bottom-up by considering the *context primitives* that can be obtained from load sensing, and the construction of higher-level context capture techniques.

The weight lab is comprised of four 'load sensitive' artefacts: the floor, two tables and a shelf. Other, non-load sensitive artefacts can be arranged on these surfaces. The arrangement of these artefacts is not fixed; all components can be moved to create new experimental configurations, for details see [Schmidt,02] and [Schmidt,02a].

In order to make the artefacts aware of the load placed upon them and also of the position of an object or a subject, the surface is augmented with load cells. Dedicated hardware has been developed to drive the load cells and facilitate the data acquisition, see Appendix B: Load Sensing System. The current prototype offers both wired RS-232 and wireless communication interfaces.

Weight Floor

The floor is constructed from a wooden structure of 240cm by 180cm. The surface of the floor is mounted on three supporting timber beams resulting in an overall height of approximately 9cm. At each corner of the floor a single load cell is mounted into the supporting timber, see Figure 11. The whole floor rests entirely on the four load cells and the remaining structure is not in contact with the conventional load bearing floor of the building.

An estimate of the typical load in our environment is: 2 people (70kg each), 2 armchairs (15kg each), a coffee table (10kg), a shelf (20kg), and the weight of the floor (80kg) resulting in 280kg. Due to the structure and anticipated loading of the floor, we selected S-load cells each with a capacity of 1000N (overload safe to 2000N).

The four load cells are connected to the data acquisition hardware. The floor system incorporates an 'auto-tare' mechanism which allows discounting the position of stationary objects. More specifically, whenever the load is considered stable this preload is stored ($F0_1$, $F0_2$, $F0_3$, and $F0_4$) then factored into successive calculations to determine the point of pressure of further objects, e.g. the position of someone is walking on the floor. Using this mechanism, furniture can be added to the floor and automatically included in the position calculation once the floor is not occupied for

Figure 11: The floor installed in the lab setting (left). Enlarged view of the load cell embedded into the floor (right)

several seconds. Note that in case when only a single person is on the floor the position of the person is determined, whereas in case of more people, the point of pressure is the centre of mass of the group. Although not yielding their individual positions, such information may still help us determine the locus of activity.

The floor recognises three types of event: no interaction, people moving and stationary occupation (e.g. someone is sitting in an armchair or standing in front of the pin board). These events can be recognised with an accuracy of over 99%. The position of a single person moving in a space already populated with furniture and covered with a rug can be acquired with approximately 10 centimetre accuracy. The accuracy is also dependent on the physical structure of the floor.

Weight Tables and Shelves

Load sensing technology is embedded in several pieces of furniture: a small coffee table, a larger dining table, and a shelf/drawer unit. Both tables are constructed using load cells installed between the table top and the frame, such that the table top rests on a load cell at each corner (see Figure 12).

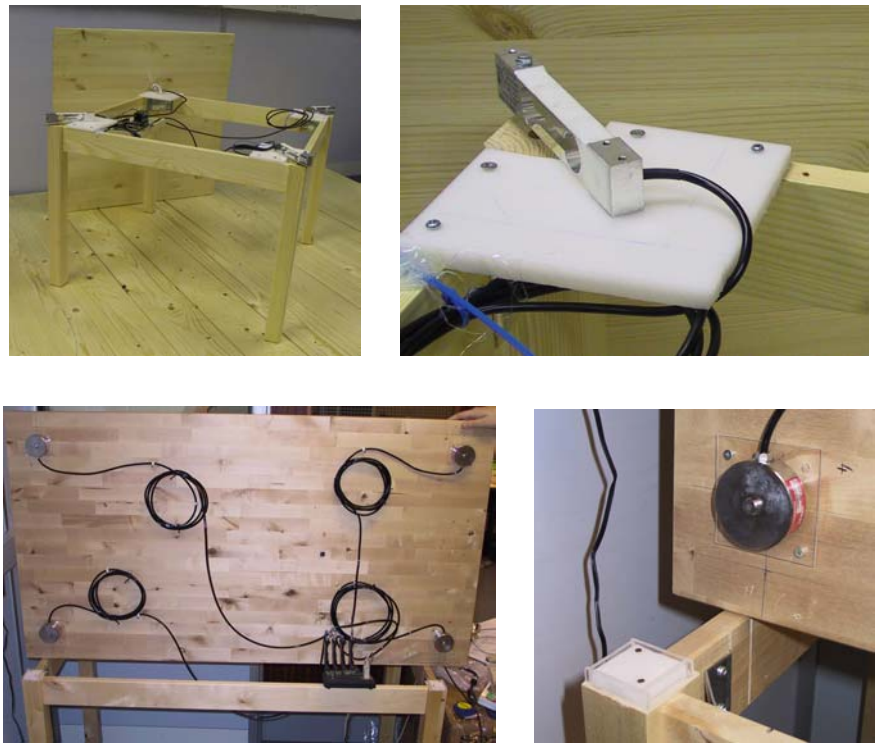


Figure 12: Coffee table (top) and dining table equipped with load cells (bottom). Close ups of the load cells and how they are fixed (right).

The coffee table can measure a maximum load of 8kg (including the 1kg table top). For the larger 'dining' table load cells with a capacity of 500N each, resulting in an overall capacity of 200kg are selected. The tables and shelves use the same data acquisition hardware as the floor, but run different software on the microcontrollers.

The shelf unit is placed on 4 load cells of the same type as the ones used for the large table. The position of the interaction is assigned to a column of the shelf, as a single set of load cells cannot detect where in the vertical axis the interaction took place. Additionally, some of the shelf boards within the unit are equipped with load sensing boards. These boards are built based on cheap consumer load cells.

4.4.2.3 Lessons learned for load sensing

Prototyping the load sensing technology and finally a complete load sensing environment was a time consuming and expensive effort, with regard to space, development cost, and technology. Nevertheless building a large system with multiple surfaces was necessary to proof the usefulness of such a technology. In particular the setup in a living lab environment, deploying full scale furniture, was essential to evaluate important properties.

The setup showed that it is possible to realise sensing and tracking technologies unobtrusively and compatibly to everyday environments. Creating the weight lab it became apparent that the basic load sensing technology introduces little complexity and allows for simple integration in the design of artefacts. And as it is a proven and robust technology the system is running with little attention since it was installed. Having this permanent setup allows data collection and different methods for evaluation of the system to be explored.

From the prototypes we learned that providing basic primitives is an important issue. In particular primitives related to the interaction with the artefact have been a useful abstraction for further developments. Some of the primitives are shown in Figure 13, using the notation suggested in [Crowley,02]. These abstractions make the use of load sensing systems more versatile.

In further projects primitives that provide information about the placement and removal of objects from surfaces have been used. One example is a retail shelf where

these primitives correlate to the user's interaction with products on the shelf. By recognising that a product the user takes out of the shelf actions, similar to recommender systems in e-commerce applications, can be taken. Other examples are in a series of designs that have been carried out by Bill Gaver and his group at the Royal College of Art in London. In this work, which originates from the cooperation in Equator, designs are created that use load sensing technology, see [EQUATOR,02].

4.5 Learning from Prototypes – Generalising the Approach

Illustrated with various prototypes in the previous sections the feasibility of realising bottom-up context-awareness is demonstrated. Each of the prototypes made one specific artefact context-aware; however each of the artefacts also reflects a type of entity. To learn and communicate from the experience of building the prototypes it is desirable to describe the result as well as the process by which that was achieved. To reuse the knowledge and also potentially parts of an implementation an efficient way to communicate details of contexts and awareness about entities, as well as about the process has to be found. This is in particular relevant within the research community to make it easier to build on one another's work. As Ubiquitous Computing and in particular context-awareness is a multi-disciplinary field the communication also has to bridge gaps between subjects.

The aim when describing context aware systems is on one hand to precisely layout the way contexts are established for a particular entity and on the other hand to formulate

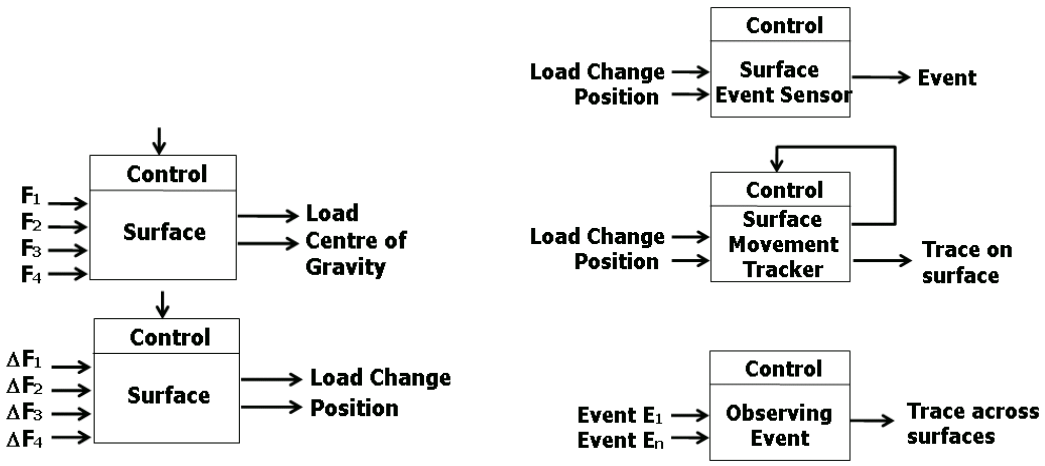


Figure 13: Load sensing primitives.

it as general as possible, so that transfer and reused for similar entities is easy. A formal way of describing it could be very powerful, however completely formalising the description seems to be not practical with the current understanding of context and awareness.

When describing context and awareness in an entity centred style, further different issues have to be included, such as

- How does the entity, and thus contexts of this entity, related to other entities?
- What are major side conditions and requirements stated by the way the entity is deployed?
- Whether and how the contexts described for the entity are restricted to a wider scenario of use?
- How do contexts relate to each other?

The quest for a way of describing context brings to mind a statement by C. Alexander [Alexander,79,p.67] where the use of patterns is motivated:

> *"If I consider my life honestly, I see that it is governed by a certain very small number of patterns of events which I take part in over and over again.*
>
> *Being in bed, having a shower, having breakfast in the kitchen, sitting in my study writing, walking in the garden, […] There are surprisingly few of these patterns of events in any one person's way of life, perhaps no more than a dozen. […] Of course, the standard patterns of events vary very much from person to person, and from culture to culture."* [Alexander,79,p.67]

Similarly it appears that the number of contexts that are relevant for a single entity is quite small; however, by combining entities in temporal and spatial ways, complex scenarios are generated. Entities and contexts related to these entities seem to be a logical extension of the patterns describing types of places in [Alexander,77], but on a smaller scale. Before describing a pattern language for context the next subsection is meant to recall the origin of pattern languages.

4.5.1 Pattern Languages

In the books “the timeless way of building” [Alexander,79] and “a pattern language” [Alexander,77] Alexander founds a language to describe buildings and architecture on a greater scale by viewing it more from a social perspective than from a constructional. The 253 patterns that are described provide a basic ‘vocabulary’ to express buildings.

> *“The people can shape buildings for themselves and have done it for centuries, by using languages which I call pattern languages. A pattern language gives each person who uses it the power to create an infinite variety of new and unique buildings, just as his ordinary language gives him the power to create an infinite variety of sentences.”* [Alexander,79,p167].

This statement also suggests that the language is already there and that by writing down the pattern language it was discovered rather than created.

Central to the pattern language is that it includes a hierarchy and also that patterns are not to be seen in isolation. The relationships between patterns are an inherent part of the language. It is also apparent that already a subset of patterns forms a language of its own.

These properties and the fact that a pattern language can be constructed using plain language so that it can easily understood without specific knowledge on a notation made it appealing for other research areas, too. It was in particular adapted to the problem of software reuse in the field of software engineering, as given in [Gamma,95]. Recently also researchers in HCI proposed the use of patterns to communicate design solutions [Borchers,01].

Understanding the use of technology in relation to context, especially in a home environment, is researched by a team at the University of Nottingham within Equator [EQUATOR,02]. Here ethnographic studies have been carried out to find typical patterns of usage, of technologies in home environments [Crabtree,01]. To communicate their findings they extended the pattern language described by Alexander with the inclusion of technologies into material arrangements [Crabtree,01a].

4.5.2 A Pattern Language to Describe Contexts and Awareness

The very basic observation by Alexander, from an architect's viewpoint, is that life is organised around space and takes place in repeating patterns of events, as pointed out in the statement "*… these patterns of events which repeat themselves are always anchored in space*" [Alexander,79,p.69]. When taking this further into the domain of context-aware systems the notion of space and place can be extended so that any entity can be an anchor, considering contexts as, or patterns of, events.

In our research so far we have identified a number of entities and their contexts; however we are far from a comprehensive vocabulary. However even starting out with these few 'words' that we have identified so far, can help to form understanding of context and context-awareness. Recognising that only a small number of aspects are addressed, there is no specific order or hierarchy to the patterns.

The pattern language proposed in this section is, like most languages, not static, and as new entities, new contexts, new methods, or new approaches arise additional concepts may be added to the language.

Each pattern has two major logical parts. In the first one the entity and the related contexts are described, it is also placed within a scenario, and relations to other entities, which are similar, are described. In the second part a particular solution of how this particular entity was made context-aware, is described.

Each pattern has a **name** and a **number**. As already pointed out in [Alexander,77], the names of the patterns are important because they create the vocabulary which is used to reference the concept described using a pattern. So far numbers are chosen sequentially with developing context-aware artefacts, rather than providing a hierarchical structure.

Then the **entity** is described. As explained earlier the entity may by a subject, an artefact, or a part of either one. The entity should be as specific as necessary but as general as possible. The entity is often strongly related to the name of the pattern. The entity is followed by a **scenario** explaining more about the entity, together with **examples** of such entities.

Derived from the scenarios a list of **contexts of interest** is stated. Here contexts that are particular to an entity are described. Contexts can incorporate what an entity does or what it is used for, the relation to its environment as well as intrinsic properties. Potentially this list can be infinite, however selecting context should be orientated on 'what is normal' and leave the 'odd cases' out. The contexts that are perceived with the implementation, described later, will be repeated there.

If applicable, it should then be stated **to who the contexts are of interest**, when this is possible to say. This may be a particular device, the entity itself, or in most cases this is open and not further specified.

Forces are the major constraints to an entity, as described in the previous chapter. In particular there should be explanations of what important side conditions are for this entity, such as robustness, weight, battery life, and unobtrusiveness.

Then **sensing technologies** which are used to acquire contexts are described. Here in particular is should be stated which sensing approach (local, distributed, homogenous, or heterogeneous), which sensing perspective (intrinsic, observed), and which physical sensors are selected. When using contexts of other entities as sensors these should be included here as well. Often it is argued that the more sensors and the more diverse they are the better perception is supported [Laerhoven,02], however here the value of the description is to point out what the important sensors are.

Then **perception techniques** which are deployed to get the contexts are specified (e.g. rules, statistics, and Neural Nets). Here in particular the learning behaviour of the systems is described (fixed, learning phase, adaptive) and motivated.

Also the **device architecture**, in particular how sensing, perception, and processing is arranged is described. Here in particular the question of how much processing power and storage is available (e.g. processor, ram) and where the processing takes place (e.g. local, in the backend) is discussed. Here the motivation for the selected device architecture should be provided.

Dependent on the device architecture and also on the partners that use context the **communication technologies** incorporated are stated. In particular the technologies

such as wireless (e.g. radio and infra-red) or wired, are described and the motivation behind the decision stated. In certain cases there may be no communication.

As an important part, an **implementation example** is given, providing more details on realisation issues. This may be accompanied by a photo, schematics, code or similar material or may contain references to these materials.

Finally **references to other patterns** that are related should be stated. Here the references should be ordered in three categories: first patterns where entities are described that are used as a sensor in the current patterns (lower level patters), second patterns that are similar to the current pattern (same level), and third patterns that are using the current pattern as a sensor (higher level patterns).

4.6 Patterns of aware artefacts

During the research that was carried out a number of different prototypes were built to develop an understanding of the problem space related to context-aware systems. Building prototypes as a methodological research approach is discussed in more detail in chapter 8.

Some of the prototypes reflect large scale projects behind them whereas others evolved from rather small projects. However theses prototypes represent certain patterns. So far the following patterns have been identified, which are included in Appendix C: Patterns (see page 51).

- Context Pattern #1, battery powered hand held electronic appliance
- Context Pattern #2, mains powered stationary appliance
- Context Pattern #3, non electronic portable every day objects
- Context Pattern #4, non electronic stationary every day objects
- Context Pattern #5, non portable furniture with horizontal surfaces
- Context Pattern #6, furniture that people sit on
- Context Pattern #7, garment
- Context Pattern #8, location awareness for mobile computing devices
- Context Pattern #9, context aware recording devices with communication

As the number of patterns, or the vocabulary, is still limited the references to other patterns are less meaningful. Therefore this is omitted in the descriptions. When the vocabulary is increased this becomes a powerful mechanism to relate knowledge.

These 9 patterns introduced above do not provide a complete pattern language for context-aware systems in Ubiquitous Computing. The first 7 patterns are closely in line with idea of bottom-up context-awareness that is tied to an entity.

Pattern #8 “location awareness for mobile computing devices” is different to some extent, because it is restricted to a certain type of context for these entities. Also the approach is not modelling the contexts of a device. Location is used as reference to access services. Nevertheless this pattern is very important, and was the focus of much research carried out so far. Pattern #9 is also different somehow as it is concerned with the very specific application of tracking. Nevertheless this pattern is in line with idea of modelling an entity, in this case a transport box. The contexts of interest however differ on the content of the box, as for certain goods the temperature is more important and for other the surrounding magnetic field.

The patterns can be extended further, either by considering new entities or by creating patterns that only look at subset of entities described in the patterns above, e.g. creating a new pattern ‘chairs’ that restrict and refines pattern #6.

However, in certain cases new entities can be reduced or at least related to patterns already described. For examples a pattern “Body Parts” including foot, hand and head can be related to pattern #7 garment, where the entity is the extrinsic observer of the body part. Another example is “Accessories” such as a handbag and a watch. These entities can be considered using the patterns #1 portable electronic device or #3 everyday objects.

4.7 Artefacts Become a Part of the Application

The approach described regards artefacts as being context-aware without targeting a specific application. However when artefacts supply context to a system, even if they are themselves non-electronical, they become a part of ‘the computer system’. When supplying context they become at the very least an input device to the system. Inherently the notion of ‘computer system’ and also of ‘application’ becomes unclear

with this background. In particular the following issues evolve from this change and when designing computer systems and user interfaces they have to be considered.

- Physical distribution of artefacts and their relation matters.
- Communication of artefacts among each other and also with the underlying system becomes central.
- The development process, building computer systems and applications, has to include artefacts and context acquisition as relevant aspects.

Artefacts and entities in the real world are inherently distributed; and their distribution is most often meaningful [Kirsh,95], [Beigl,00]. Taking this into account and providing a platform that supports communication and also recognises the semantic of distribution, is essential to efficiently support the development process. In chapter 6 these issues are further investigated.

Making artefacts aware and providing context information to a system also has severe implications on human computer interaction. The balance between invisible computing and meaningful interaction poses particular further questioning. These issues are discussed further in chapter 7.

4.8 Discussion

The work presented here suggests that it is useful to anchor context at entities, such as subjects and artefacts. Starting out from a specific entity the central question arises: **What is a generic set of intrinsic properties or contexts for this particular entity?** The question can be made more fundamental, is there such a generic set?

For most artefacts that have been investigated in the research carried out it appears to be manageable to find a set of properties that cover the most common situations. However the attempt to model all possible situations an artefact may encounter will result in a model of the world for each object. In this bottom-up approach one of the most difficult decisions is when to stop modelling and to what degree of detail the model should go.

The relationship between artefacts, which is essential to the understanding of situations in many cases, is also a central issue when modelling. Modelling relationships directly in entities can lead to fairly complex models as the number of entities increases. Here generalising the relationship is helpful. Instead of modelling the relationships between two specific entities it is simpler to model the relationship between a specific entity and an abstraction of other entities, e.g. modelling the relationship of a cup on a horizontal surface, instead of many relationships between a cup and a specific table. Similarly when this is modelled for the table (e.g. an object placed on the table instead of specifically a cup on the table) using correlation over time can easily reveal a specific relationship.

When implementing and prototyping aware artefacts or context acquisition systems many different decisions are taken, e.g. what is sensed, where is it sensed, who is benefiting from the context, which situations are regarded and which are not considered, and how is communication facilitated. Making these decisions explicit and consciously knowing the alternatives optimises the development process and increases the understanding of the system design.

As advocated in this chapter prototyping is a central way of understanding context-aware systems and their use in real world environments. In most of the projects mentioned central aspects and applications emerged by engineering a solution and deploying a system in a real world environment [Schmidt,00], [Schmidt,02a]. However prototyping, especially building functional physical prototypes, is a time and resource consuming task. The lessons learned from one prototype will help to build the next one. In many labs this knowledge is bound directly to people as such knowledge is difficult to communicate. The use of patterns as a less formal way of description is helpful especially when sharing experience with other disciplines (e.g. design and ethnography).

4.9 Summary and Conclusion

In this chapter the concept of bottom-up context-awareness has been introduced. The central idea is that context is always connected to an entity. This concept provides means to structure context aware systems in a natural way. The design space for this

way of modelling is described and the basic approach is explained. Closely related to bottom-up modelling is the method of prototyping systems.

After providing a short overview of prototypes that have been built over recent years two of them are presented in more detail. The report on prototyping a context-aware mobile phone provides insight into the method. The development of the weight lab explains how experiments are carried out to understand the basic context acquisition capabilities of systems and how they can be integrated in everyday environments.

The experience reported shows that aware artefacts are a useful abstraction and can act as a foundation for further applications. Looking at the realisation a close relationship between the conceptual model and the actual implementation can be observed. This close relationship makes it simpler to understand systems and also opens ways for extensibility of systems. Furthermore it also supports the composition of systems.

A main value of building functional prototypes is that during this process a great understanding, of specific technologies and the whole system, is gained. Creating prototypes often reveals side issues that have not been considered before. Chance inventions and side findings are often triggered by building prototypes.

Prototyping is expensive in every sense; nevertheless from the work reported here it appears that there is no real alternative. Simulations, such as Ubiwise [Vijayraghavan,01], are only applicable when the domain is well understood. Especially in the domain of sensor based context acquisition, physical prototypes seem to be the only method to collect real data. To ease this problem methods and tools for prototyping of context acquisition systems is a central concern. In the next chapter this will be addressed introducing a rapid prototyping platform.

Chapter 5

Supporting the Development and Tools for Rapid Prototyping

Context acquisition is a central concern in Ubiquitous Computing as it is the prerequisite for all context-aware systems and applications. In this chapter it is assessed how context acquisition can be efficiently supported by tools and methods. Acquiring and providing context is crossing fields such as device and hardware design, system and perception software, as well as communication. Providing efficient development support is therefore difficult and still in many ways an unsolved issue.

5.1 Analysis: Libraries and Tools are a Necessity

In previous chapters an overview of sensing technologies and a selection of prototypes of aware artefacts are presented. Looking at the projects described and at the patterns identified it becomes obvious that certain sensor, perception, software, hardware, and communication problems are solved over and over again. Looking at the implementations the wheel is re-invented slightly differently in each system.

As the process of creating a system that comprises of hardware, software, and communication, system design is time consuming and also expensive. It is appealing to provide tools and methods to reuse knowledge and also to automate steps in the process. The concept of informal reuse can be seen when looking at publications and

demonstrators from institutions where there is an extensive experience in a certain platform. It is apparent that a basic design will reappear in different projects. Developers build up their building blocks which are then used again to save development time. This knowledge is often very much tied to a particular developer.

Tools and methods to support the implementation of sensor based context acquisition systems can be deployed in various ways. It ranges from decision support when selecting sensing technologies for a particular context, ready-made rapid prototyping systems for context acquisition, libraries and building blocks, to an integrated development environment that can support design, hardware, communication and software development.

5.1.1 Software Libraries and Hardware Building Blocks

A general approach in computer science is to provide libraries that ease the development of applications in a certain domain. Functions and procedures that are repeatedly needed are developed once and shared and reused in following developments. Software libraries however are most often bound to a particular operating system or execution environment assuming an underling software system and a basic hardware configuration.

When providing software libraries for context acquisition systems the issue becomes wider as there is no standard sensor system or sensing platform. Especially the software (comparable to system drivers in conversional systems) that realises interfacing with the actual sensors is mostly dependent on the sensing system. Here libraries and building blocks can be provided to make the development of hardware related code easier. Furthermore, in context acquisition systems communication with potential context consumers is a major issue. In many cases this also includes the distribution of data. Libraries that facilitate communication between sensing devices and other devices and also with the backend infrastructure are useful to make this process more efficient.

As context acquisition systems include sensors this often involves the design and implementation of specific hardware using a particular set of sensors arranged in a specific way to suit the task. Providing basic sensor building blocks to speed up the process of hardware design becomes an obvious goal. Similarly processing and

communication building blocks are also helpful for these functional units. It is suggested that a variety of libraries for different platforms (e.g. embedded systems and PCs) and device building blocks (e.g. schematics, board layouts, and physical designs) can improve and hence ease the development process.

5.1.2 Context Acquisition Design Method

When designing a context acquisition system one of the first steps is to select the right sensors, algorithms, processing cores, energy supplies, and communication technologies. The decisions made here have far reaching consequences for the context acquisition system.

At this step it is central to have knowledge about what sensors and algorithms can contribute to get information about a particular context. The selected sensors and algorithms have implications on the processing system, energy consumption, and what type of communication is required. As for a particular context there is often more than one sensor that can contribute and there is also different cost associated with different solutions.

Given that the knowledge about architectures, building blocks and libraries is available a method to guide trough the design of such systems can be of significant value. Having a method this is then a starting point to develop further tools that support particular parts of the design process.

5.1.3 Ready-Made Deployable Rapid Prototyping Devices

Additionally, devices that offer the most common sensing opportunities built-in, combined with basic processing capabilities and communication can help to get a system deployed and explored in very short time.

> **Hypothesis 4: A rapid prototyping platform to make artefacts context-aware can be provided. Such a platform will simplify and speed up the prototyping process of such systems.**

The advantage of a hardware platform is that creating a specific context acquisition system only requires the development of context specific software on the sensor devices and potentially in the backend. As the set of sensors is defined by the

hardware implementation also efficient software libraries and contexts acquisition software can be supplied.

A prerequisite for rapid prototyping systems for context awareness is that they are made in a way that they can be attached, included, embedded, or integrated with whatever artefact should become context-aware. This is a real challenge for the physical design of such tools – especially size and power consumption become a major issue.

As it is impractical (or even impossible) to include all sensors that may be used in one sensor board it is important that a simple way of attaching and detaching sensors is included in the design. As it is not likely to foresee all sensors developers want to use, ways for extension have to be included.

All three paths – libraries, development method and tools, and a rapid prototyping platform – have been explored in the research that led to this thesis. In the following sections the main issues in each of the approaches are presented.

5.2 Context Acquisition Libraries

When designing a library for context acquisition the main problem that appears is that there is no standard sensor platform. Furthermore, when looking at the patterns presented in the last chapter it becomes apparent that in many cases context acquisition systems have to be designed from scratch to be useful beyond an early prototype stage. The physical design and the host artefacts must be taken into account. This process of designing whole context acquisition systems rather than providing software for a given platform sets new challenges for the provision of a library. It includes system architectures and hardware issues and is also concerned with communication, protocols, and software. The libraries can not be viewed in isolation. System architecture, hardware platform, and communication are dependent on each other and also influence the software deployed.

The libraries presented in this section reflect the experience gained from implementing various prototypes. The hardware building blocks and software components evolved by factoring out common parts of a union of all implementations carried out in different projects. An overview of the library is given in Table 10.

Context Acquisition Library Structure		
Category	*Sub Categories*	*Implementation*
Architectural Frameworks	▪ Attached sensing architecture. ▪ Wireless single consumer architecture. ▪ General wireless sensing architecture	System architectures
Hardware Library	▪ Processing cores and memory units. ▪ Sensor blocks ▪ Communication blocks ▪ Power supply blocks.	EAGLE CAD files
Software Library	▪ Program Templates	Program skleletons in C and function in PIC-C
	▪ Sensor drivers ▪ Communication drivers	Drivers implemented in functions (PIC-C)
Perception Library	▪ Statistical functions ▪ Time domain analyis	Function in PIC-C
Backend Library	▪ Serial line access ▪ Network access ▪ FuzzySpace access	Variety of skeletons and functions/classes in Java, C/C++, and Visual Basic for Linux and Win32.

Table 10: Context Acquisition Library.

5.2.1 Architectural Frameworks

The overall architecture of a system has implications on the hardware, software, and communication. The following three general architectures are suggested as framework when building context-aware systems.

- **Attached Sensing Architecture**. This architecture is used when context acquisition is attached to the context consumer directly. A single sensor system is physically built into or onto a device that makes use of context. The connection between sensor system and device is wired (e.g. using a serial line). Examples are sensor boards connected to handheld devices or phones [TEA,98]. See Figure 14 (top).

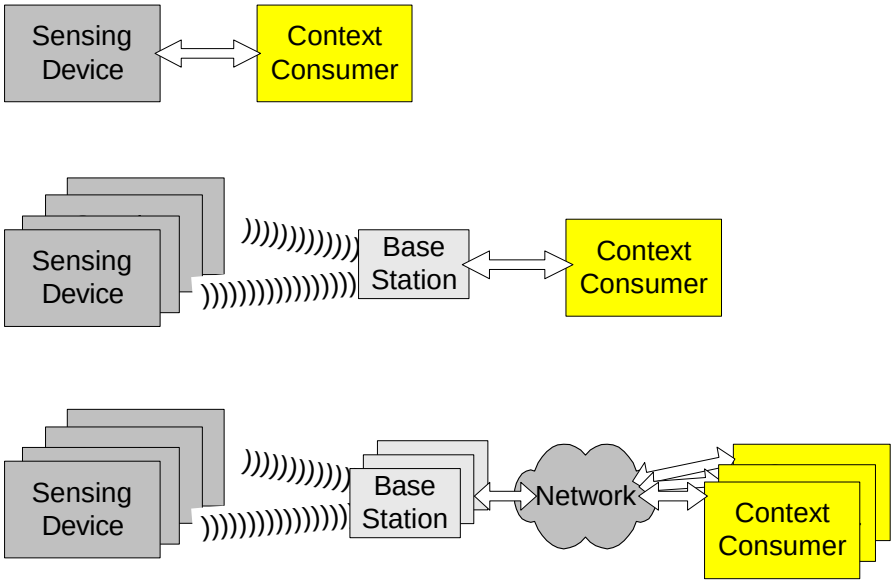


Figure 14: Basic System Architectures.

- **Wireless Single Consumer Architecture**. The sensing system consists of one or more sensing nodes which are wirelessly connected to a base station. The base station is connected via wires to the device that is making use of context. The sensor system can be distributed. See Figure 14 (middle).

- **General Wireless Sensing Architecture**. The sensing system consists of one or more sensing nodes which are connected to one or more base stations. The base stations are connected to the network or more specifically to a distribution platform (e.g., a FuzzySpace see chapter 6). All applications that make use of context are also connected to the network or distribution platform. See Figure 14 (bottom).

For each of the architectures a basic hardware framework, software framework, and communication setup is be provided. These frameworks are then filled with building blocks as outlined in the following sections.

5.2.2 Hardware Library

The requirements on the design of devices that provide context are manifold. In many cases only an individual design, that is targeted to the artefact which should become context-aware, offers a usable solution.

Hardware libraries are a possible solution. In conventional electronic CAD systems libraries with a wide range of components are included. These components can then be added to schematics and PCB designs. Mostly these components are single devices, such as resistors, capacitors, connectors, and processors. In most cases a schematic (and also the PCB) consists of a number of components connected together. The connections are made according to the datasheets of theses components and using general knowledge about hardware design. This is a time consuming process.

The approach with the hardware library is to provide larger building blocks to speed up the design process. The library developed is implemented as EAGLE schematic and PCB layout [CadSoft,02]. The building blocks provided are designed and documented in a way that the developer can 'copy&paste' them into his designs.

5.2.2.1 Processing Cores and Memory Units

For all units in the system, either sensing units or communication units, a processing core is required. The processing core includes a microcontroller with a number of I/O pins. The building block is designed in a way that all I/O lines can be used when power is connected without further components. In some cases the core is a single component (e.g. using a PIC16F628) but typically this includes the MCU together with an oscillation circuit and a reset mechanism. Also one unit with high resolution A/D conversion is included in the library.

For an overview of the processing core library see Appendix D.1: Hardware Building Blocks HWcore.

The storage in the microcontrollers used is very limited, therefore additional memory may be required. Therefore a 64Kbit FRAM chip is included in the library, too. The memory unit (containing the FRAM chip and pull-up resistors) is connected via I2C and offers a non-volatile storage at minimum energy consumption.

5.2.2.2 Sensor Blocks

In chapter 3 different types of sensors and their use for context acquisition is discussed. This library contains building blocks for various sensors. As many sensors can not directly be attached to a microcontroller the building blocks also include the signal conditioning circuits.

Each building block in the library is created so that it can be connected directly to power (typically 3 or 5V) and the output of the sensor can directly be connected to the MCU. Dependent on the sensors this may be one or more analog pins, one or more digital input pin, or a more complex communication interface such as SPI or I2C.

The library deploys a naming schema, where AOUT* is a analog output of the sensor which should be connected to an analog input of the processing unit, DOUT* is a digital output of the sensor and should be connected to an digital I/O pin, DIN* is a digital input pin to manipulate the sensor and should also be connected to a digital I/O pin. Names ending with *OP indicate that these pins are optional – so they can be connected but it also will work if they are left open.

On overview of the library is given in Appendix D.2: Sensor Building Blocks HWsensor.

5.2.2.3 Communication

In all cases where context acquisition systems are created communication is required. By choosing a particular architectural framework the basic communication architecture is specified.

In the simplest case this is a wired connection between the processing core and a device that accommodates the context consumer. For these links the library offers designs for serial line communication (RS232) and I2C connectivity.

The library also includes different types of wireless units. As AM and FM transmitter and receiver have different parameters, especially regarding communication speed and energy consumption various building blocks are provided. Additionally to transmitters and receivers also transceivers are included. Building blocks also contain an antenna (e.g a PCB track) which size is depending on the frequency of the module. A further option is a wireless link based on infrared transmitters and receivers.

All communication building blocks in this library are designed so that they can be directly interfaced to a MCU and accept serial data that is appropriately coded and not exceeding the speed specified, see also the software library description later.

The library is presented in Appendix D.3: Communication Building Blocks HWcomm.

There are further collections of hardware building blocks, such as the power supply library, which are useful to rapidly build systems.

5.2.3 Software library

The software libraries are designed to ease the development of sensing and context acquisition systems. In particular the functions support programming of systems which are running hardware that is constructed using the building blocks from the above hardware libraries.

The library includes program templates that relate to the building blocks, functions that provide hardware drivers for sensing and communication, and functions implementing basic perception. These software parts are all implemented in C, written for the CCS-PCM compiler [CCS,02]. Additionally, the library also contains backend functions to access the context acquisition system; those are available in various languages and for different PC platforms (e.g. C, Visual BASIC, and Java).

5.2.3.1 Program Templates

For each of the cores specified in the HWcore-library a program template is provided. The template is divided in two parts. A general c-file that will be used as starting point by the developer and will be extended with further functionality is one part. And a C-file that implements a driver for a particular core is the second[4].

Similarly for the additional memory also a C-driver file to include is provided. It offers basic read and write access to particular memory addresses. It can be extended with functionality to store typed values in the memory.

An overview of core-drivers and memory drivers is listed in Appendix D.4: Core Libraries SWcore.

[4] The CCS compiler used does not support precompiled modules as usually known from standard C development. Before each compiler-run the main file is extended with all includes to one single file which is then compiled. This makes it necessary to include the c-code of functions.

5.2.3.2 Sensor Drivers

For each sensor building block a driver, realised as a C-file, is provided. The driver file provides access functions to the sensor values. This is implemented in different ways depending on the sensor and how the sensor is connected to the core. Before including a sensor driver in the code it must be defined to what pins the pins of the sensor is connected. This is realised using a set of 'define'-commands. The 'define'-commands required are specified in the header of the driver file and if appropriate default values are set. See Appendix D.5: Sensor Drivers SWsensor.

5.2.3.3 Communication Drivers

Different communication drivers and functions are provided for the building blocks specified. This makes the implementation of communication, in particular wireless communication, easier. Similarly to the sensor drivers the connections between the communication units and the core have to be described using 'define'-commands. Also the speed for a connection has to be specified using a 'define'-statement.

As the complier offers already functionality for wired serial communication and wired I2C communication, the library is mainly concerned with wireless communication. The library offers a 'printf'-function, which is similar to the RS232-'printf'-function provided by the compiler, for sending data. On the receiver side two types of functions are implemented, one actively polling and another one using carrier detect (CD) to trigger an interrupt. In particular these functions realise error detection using a CRC-based mechanism.

As wireless communication is a main energy consumer in many designs also functions to power down these components are implemented if supported by the hardware.

An overview of functions offered in the communication library is given in Appendix D.6: Communication Drivers SWcomm.

5.2.3.4 Perception Library

In the perception library (SWperception) algorithms for cue and feature calculation are collected. The implementation reflects some of the perceptions methods discussed

earlier. In particular functions that are light weight, such as basic statistics and time domain analysis are included.

The functions which operate on a vector of data are implemented in two versions; one where the data is held in a global array and the other were data is passed through parameters. Using a global array seems like poor software engineering practice, however, given the minimal storage and the speed of the MCUs it can save memory and copy operation. Given the constraints of the MCU the functions working on a global buffer are included in the library.

5.2.3.5 Backend Software Libraries

Recalling the three basic architectures presented above it can be seen that the context consumer has to communicate in one of the following ways:

- accesses directly the sensing device,
- accesses directly the base station, or
- accessing the context acquisition system indirectly via the network

In the first two cases a standard way to do it is to use the serial line (or USB which emulates a serial line). In the third option the access is done using UDP or HTTP over an IP network (which can be encapsulated in a high level library such as a distribution platform).

The backend library caters for those cases. The implementation for Visual BASIC uses the Microsoft Comm-control. The C implementation for a Win32 and Linux uses native API calls to provide the functionality. The java version uses the javax.comm package that is available for different platforms.

5.3 Context Acquisition Design Method and Tool support

To support the design process of context acquisition systems it is useful to assess what steps are usually accomplished in this process. In particular it is important to identify at which point decisions are taken. Furthermore it is important to recognise how these decisions influence the process further. These observations lead to a description of the

design space. Based on this knowledge it becomes possible to create tools that provide advice and support in this process.

5.3.1 Design Steps and Decisions

The following steps reflect on the design process of a context acquisition device as done in a number of projects and also reported by other researchers. The steps outlined can also be regarded as method to build context acquisition systems. In Table 11 the method is summarised.

5.3.1.1 Method

The method outlined here provides a general overview on the central steps and design decisions made when designing context acquisition systems. Depending on domains further intermediate steps may be introduced or steps may be omitted because of pre-specified constraints.

Step 1. Identifying contexts that matter; checking whether or not context matters at all.

In a first step the usage of the application or the artefact that should become context-aware is analyzed. If it can be concluded from the following questions that the situation does not matter it is probably not worthwhile to use context.

A method for designing context acquisition systems	
Step	*Action*
1	Identifying contexts that matter.
2	Determine variables that distinguish contexts selected.
3	Finding appropriate Sensors to cover all variables at minimum cost.
3a (optional)	Building and assessing a prototypical sensing device.
4	Selecting recognition and abstraction technologies.
5	Specification of a context acquisition system.
6	Build applications.

Table 11: Steps for the design of context acquisition systems.

- Is the application or artefact used in changing situations?
- Vary the expectations of the user towards the application or artefact with the situation?
- Is the human computer interaction pattern different in various situations?
- Does the user's interaction with the real world environment offer information that is valuable to the application?

If it can be concluded that context may provide additional information all situations that matter are identified. For these situations the conditions of the informational, physical and social environment are identified. Real world situations that should be treated the same by the application, are grouped into one context that is named.

Step 2. Determine variables that distinguish contexts selected.

By considering possible real world situations that fall under a context variables that discriminate the context are identified. Such variables may be of informational, physical and social nature (e.g. time of day, location, number of messages, temperature, number of people in the vicinity, relationship with people near by, interaction with the device, etc.). For the minimal set of variables that make it possible to discriminate the selected context the values (e.g. ranges, sets) for variables are specified.

Step 3. Finding appropriate Sensors to cover all variables at minimum cost.

For the variables identified in step 2 possible sensors are identified. When considering sensors the following points are taken into account:

- accuracy of the sensor in relation to the variable
- the cost to provide the information (see for a cost assessment below)
- feasibility and appropriateness of using a certain sensor in a certain domain.

The resulting selection of sensors should be done in a way that the sensors cover all variables with sufficient accuracy at a minimal cost. The selection of sensors at this stage is often done based on the experience of the developer or based on datasheets and lab environment tests of sensors.

Given the knowledge about the sensors to use and the side conditions for the context acquisitions system the physical layout can be specified.

Variant: Step 3a. Building and assessing a prototypical sensing device.

On common approach is also to create after step 3 a physical sensing device using the actual sensors and providing the raw data for analysis. This intermediate step has to be taken when there is little knowledge about the sensors in question or the contexts to recognise.

Based on the decision on sensors a prototypical sensing device, e.g. sensors attached to a board in a similar form factor as the actual device and connected to data storage on a standard computer is created. Here it is especially interesting to experiment with the positions of the sensors on the device. Then the sensing device is used in the situations that should be detected and data is recorded. In a next step the recorded data is analyzed (e.g. statistics, clustering) to identify whether the raw data differs significantly for the different situations or not. If the data differs not significantly different sensors have to be selected or even different contexts have to be identified or in some case it may turn out that it is not feasible to recognise the contexts at all.

Step 4. Selecting recognition and abstraction technologies.

Based on knowledge or on data recordings available a set of cues is identified that provides a reduction of the amount of data but does not decrease the knowledge about the context. Features and algorithms are chosen in a way that all variables can be calculated from the data that is obtained from the sensors selected.

In an optimal case algorithms are found that do not reduce the knowledge about the situation with regard to the variables of interest. In this step also side conditions, especially with regards to processing and bandwidth are taken into account.

Step 5. Specification of a context acquisition system

After all components, the physical design and the algorithms are assessed the overall architecture of the system is determined. In particular the question is a single sensing device used (e.g. often in mobile systems) or is a distributed collection of sensing devices used (common in embedded settings) has to be addressed. This also includes decisions on the distribution with respect to sensors as well as processing. This is mainly to determine the overall architecture of the system. A decision could be that the sensing device is integrated in the environment and the contexts recognised are communicated to a mobile device. Communication issues and how the sensing system is connected to potential context consumers (in a distributed setting) have to be defined, too. In this step also the decision how context is provided to the context consumer such a particular application is made, typically by specifying an API.

Step 6. Build applications.

When context information is available the final step is to build applications on top of the context-aware artefact and environments that make use of context.

5.3.1.2 Cost Function

When selecting sensors and algorithms the cost is a major issue. Cost in these terms depends much on the type of artefact or application (e.g. mobile, stationary) that is to be build, the anticipated usage scenarios, and the potential user group. The issues that are taken into account reflect the requirements discussed earlier in section 3.4. Specifically the restrictions on the overall design resulting from the sensors is a major point, as well as power consumption, robustness, reliability, social acceptance, and additional monetary cost.

A cost function is highly dependent on the side conditions that are given for a specific system. Depending on the issues that are important for a specific realization these cost factors have to be weighted according to their importance in the problem domain. However, for most sensors and algorithms it is possible to describe their relative cost in certain areas such as size, weight, and power consumption.

The knowledge accumulated in the building blocks, libraries, and in the method is also used to follow the orthogonal path of providing a rapid prototyping platform.

5.4 A Rapid Prototyping Platform for Context Acquisition

Hardware building blocks, tools, and libraries offer great flexibility as outlined above, however it requires in each project the development of a new hardware platform. The value of this approach is undisputed, however it still is very expensive compared to the development of software. Especially in early phases in a research project where the feasibility of the detection of a certain context has to be assessed a different approach is often more useful. Instead of instantly designing a new hardware platform first the feasibility is assessed using a rapid prototyping platform for context acquisition. When the initial trials are successful and it can be shown that certain contexts can be detected and are useful for applications it can be moved on to the design of a specific platform using the building blocks and tools.

The method introduced above mentions the optional step 3a “Building and assessing a prototypical sensing device”. In domains where there is little knowledge of how sensor data for particular contexts looks like, it is hard to select the right sensors. To find out what sensors can provide a significant contribution towards the variables of interest deploying sensing devices and recording cues and features calculated from the sensor data is a useful way to go. For this step it is very helpful to a have ready-made deployable rapid prototyping platform available. Even if these devices do not optimally integrate with the design of the artefact that is made context aware it offers a way to gain essential real world data.

In cases where physical size and power consumption are minor issues (e.g. pattern #2, pattern #4, pattern #5, and pattern #6) providing a ready-made base platform makes the development process much simpler. In this case the base platform can include processing, memory, and communication. The development is then reduced to the connection of sensors to this base platform. Within the European Project Smart-Its these issues are addressed.

5.4.1 The Smart-Its Idea

Creating a platform that eases the development and deployment of context-aware artefacts is at the centre of the project Smart-Its. The basic idea is to build a platform which offers sensing, processing, and wireless communication that can be attached to artefacts to make them context-aware. The interest of the project is beyond the context represented by a single node. One of the hypotheses is that bringing context information together from various context sources results in collective awareness, which offers more information than the sum of all individual context sources taken into account.

The approach can be characterised by creating a device family rather than a specific device. The devices have the following properties in common:

- sensing,
- processing,
- wireless communication, and
- configurability or programmability.

As the vision is to augment everyday artefacts it is central that the technology is realised in a way that allows to post-hoc attach “Smart-Its” to these objects. The metaphor is that “Smart-Its” can be attached to objects like “Post-Its”, and provide then context-awareness for the device they are attached to. This is a realization of the concept of bottom-up context awareness.

Within the family different branches are assessed. Also within each branch an evolution can be observed resulting in various family members. The following branches evolved within the project:

- **TecO-Smart-It**
  The main design objective is to minimise the physical size and the power consumption of the devices. For more details see [Teco,02].

- **ETH-Smart-It**
  The focus in the development of this branch is interoperability with consumer devices. The communication is therefore based on Bluetooth, see [ETH,02].

- **Lancaster-Smart-It**
  In this branch the main goal is to provide a simple to use rapid prototyping platform that can be easily extended with minimal learning effort.

In the reminder of the chapter the Lancaster Smart-It is described in more detail, especially providing an insight into the design rational and the resulting realization.

5.4.2 Lancaster Smart-It Family

When designing the platform simplicity is a high priority while maintaining maximal flexibility. In particular the core hardware design is done with minimal complexity using widely available components. A further objective is that the design and operation basics of the components used should be understandable with little effort by students. Also from the experiences in educational use a further requirement evolved: the electrical design should be very robust. Similarly the core software for the complete system should be easy understandable and extensible. The hardware and software should be quickly and easily reproducible. One target audience is students and another is researchers who need a rapid prototyping tool for a proof of concept. All information about the design should be free and open.

On the other hand the system should not neglect the basic Smart-Its idea. The physical design should be compact and robust. Extensions, in particular the inclusion of new sensors should be easy – physically, electrically and in terms of software. Also the system should offer wireless operation. To speed up development time support for debugging distributed applications should be provided. Finally, the system should be mass-deployable; especially the price of individual units should be significantly cheaper than a wireless PDA.

5.4.2.1 Rapid Prototyping System Architecture

The design concept evolved from the requirements stated above. The basic functionality for each unit – independent of its function (e.g. sensing node or base station) – is incorporated into the core board. The core board offers processing, non-

volatile memory, power, and wired as well as wireless communication. It also has a physical connector for Add-On boards, offering analog inputs, digital I/O, and I2C communication. All specific functionality, in particular sensing, is realised as an Add-On board that can be physically and electrically attached to the core. Figure 15 shows a core board with a sensor board attached.

The software is also tied to these hardware modules. Specifically the software framework for the core board offers primitives for the base functionality and also to access the Add-On boards.

In the following sections the implementation of the core board, several Add-On boards and the back-end integration is described.

5.4.2.2 Core Board

The basic design of the core board is centred on a PIC16F876[5] Microcontroller offering processing, analog inputs, and digital I/O. A Ramtron FM24C64 ferro-electric memory is connected via I2C to the MCU [Ramtron,02]. The FRAM-chip provides 8KByte of non-volatile memory. These components are the only two parts in the core board that hold state information. Therefore they are not directly soldered in the board; they are hold in a socket. By physically moving the MCU and the FRAM from one board to another all program and data memory is moved. This makes it easier to keep old versions – instead of reprogramming the system or storing the whole system, only the MCU and the FRAM have to be stored for a particular version of a prototype. Furthermore the ability to transfer the processor with the software and the memory chip to another board also eases debugging.

Wireless communication, another Smart-Its key requirements, is realised using a BIM2 transceiver module from Radiometrix [Radiometrix,02]. The transceiver offers raw data rates up to 160KBits/s in half duplex mode. Running at higher data rate the module requires a balanced code (50% “0” and 50% “1”), however at lower data rates this is less critical and a serial data stream can be sent directly. As the wireless transmission is unreliable securing the transmission in software is inevitable.

[5] The board is also compatible with the PIC18F252 microcontroller which is available since 2002.

Directly on the core board there is also a RS232 connection included. To ease the connection to PC or PDAs and hence debugging a SUB-D connector is included, despite the size of the connector. The TTL-signal on the MCU is converted into a RS232 signal using converter. In the design the MAX233, which does not require external capacitors was favoured over the MAX232 to reduce the component count on the board.

As the core board should offer a flexible platform – for mobile as well as stationary use – various ways for providing power are included. For mobile use the power can be provided by 4 AAA-rechargeable or by 4 AAA-batteries. In stationary use there is also a further option to provide power externally by a supply that outputs 6-18V DC. If the power is provided by batteries or by an external source a power regular (low voltage drop version) provides a 5V supply. In case of rechargeable the 4.8V are used directly. The type of power supply can be selected by a dip-switch. To ensure that the device is not damaged in case the power is provided with wrong polarity a diode and a fuse are included.

A connector with 20 pins provides the electrical and physical extension to the core board. Add-on boards can be pluged into the connector. The connector has three groups of pins: one for I2C, one for digital I/O, and one for analog inputs. In summary the 9+8+3 connector provides 3x GND, 2x 5V-power, 5x analog-input, 7x digital I/O, one digital output and a I2C connection (2 pins, SDA and SCL).

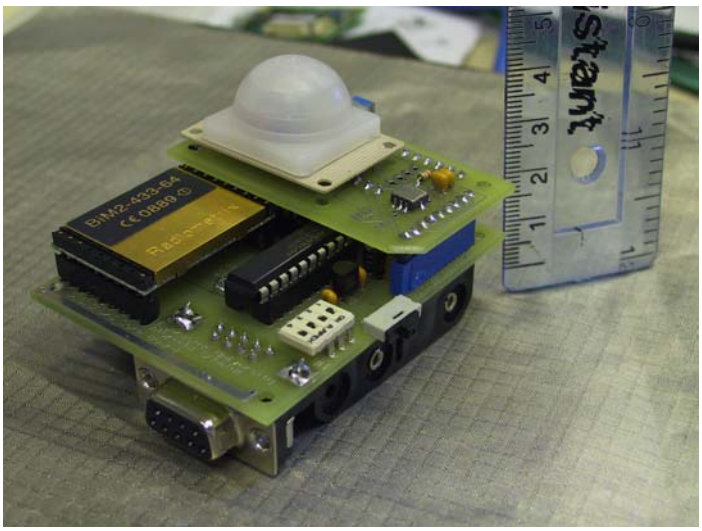


Figure 15: Core module with a sensor board attached.

The overall design results in a device with about 30 components. To understand the basics of operation it requires only looking at 5 datasheets (Microcontroller PIC16F876, RF Transceiver BIM2, Serial line driver MAX2322, Memory FR16C64, and voltage regulator 78L05). The physical size is about 55x70x29 mm and about 58 grams (110 g with 4 AAA-batteries). In Figure 16 the block diagram is shown; in Figure 17 (right) the module is depicted.

Exclusively for mobile use a further version of the core board which is much smaller 45x50x19 mm and only weights 24 grams (29 grams with a 3V battery) is developed, see Figure 17 (left). It has the same main components (processor, FRAM, and RF transceiver) and offers the same 20-pin Add-On connector. It is software compatible to the other version. The differences are in the power supply – it runs from a single 3V Lithium battery. Also RS232 level conversion and the 9-Pin Sub-D connector are omitted.

The design rational for the PIC microcontroller was the manifold. One important reason was that the MCU is widely used in education and therefore a lot of

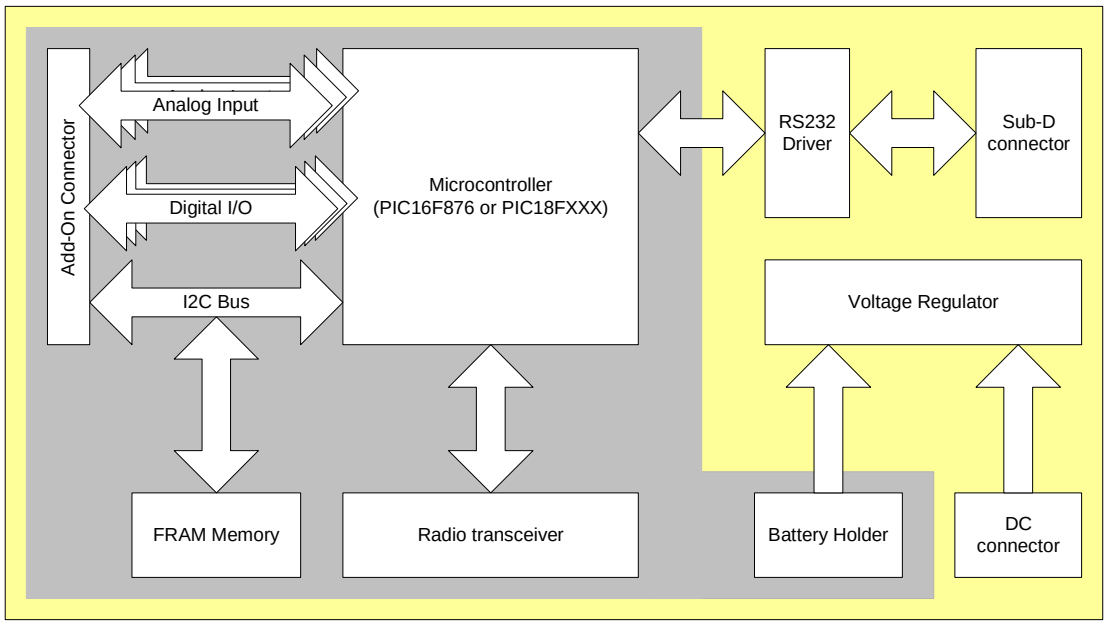


Figure 16: Block diagram of the Smart-Its core. The overall diagram shows the larger general purpose version. The part with the grey background shows the minimised version.

information is easily available. Compared to other microcontrollers (e.g. ATmega103) it is fairly cheap and power consumption is low however it offers less memory. Furthermore the availability of a compiler (which is commercial) that provides a large set of libraries was another reason for choosing this MCU. To compensate for the lack of memory the cheap and power saving FRAM chip is included in the design.

To make the hardware easily reproducible it was opted for a conversional design using no SMD technology. This decision resulted in a larger physical size however it eases assembly, soldering and debugging dramatically. Also to ease debugging 2 LEDs have been included in the design – on which is permanently on indicating the availability of power and the other which can be controlled by software.

For the core board a library and software templates are provided. The library provided functions to access all functionally offered by the core board. In particular for

- sending and receiving messages using wired or wireless communication,
- storing data in and retrieving data from the external memory,
- accessing analog input,
- reading and writing digital I/O, and
- switching the LED.

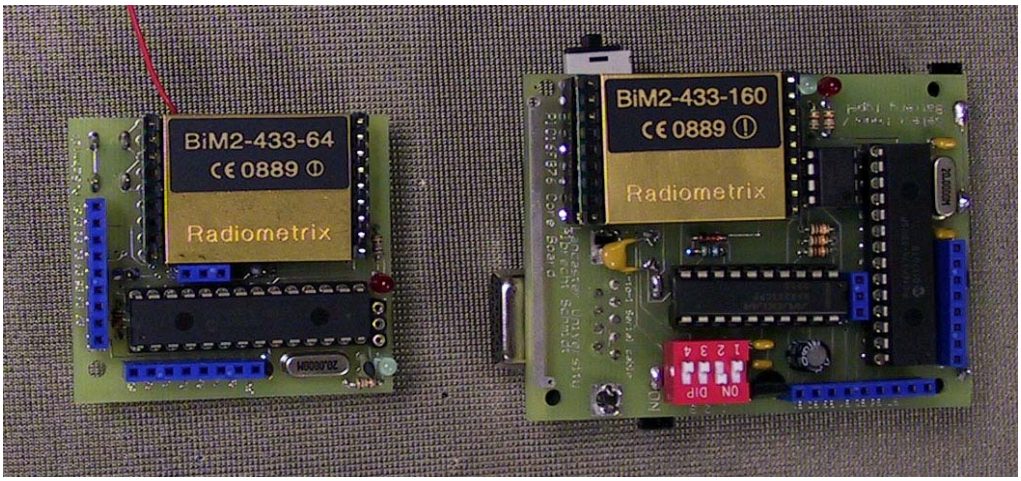


Figure 17: Smart-Its Core Boards.

The templates are programs that can be compiled and run on the core board. They make use of the libraries and provide a particular function. They are intended to be used as starting point for new development. Examples for templates are

- base-station template
  This program reads from the wireless connection and hands on everything that was received to the serial RS232 connection. This template is a starting point for developing code for modules that provided base-station functionality.

- Sensing loop template
  This program contains a infinite loop in which all analog inputs are read and then send over the wireless link as well as over the serial line. This template is the starting point for developments that read sensor data, process them, and provide them to other components in the system.

The schematics are depicted in Appendix E.1: Core Board Schematic. and Appendix E.2: Mini Core Board Schematic. Further technical details, in particular an electronic version of the schematic, the PCB-layout, and the header file are available at [Schmidt,02b].

5.4.2.3 General Sensor Board

The general sensor board is designed as an Add-On board to the core providing a basic sensing system. It is intended for prototyping of context-aware rooms and artefacts, especially in an early proof of context phase. The included sensors make it easy to collect data in situ on a real artefact or in a real world setting. The data can then be used to find out which of the sensors are useful in the setting investigated. Based on these initial experiments a specific device or Add-On can be developed.

The block diagram of the sensor Add-On is depicted in Figure 18 (left). Specifically the following sensors are included:

- Two light sensors. The boards can be equipped with a variety of pin compatible light sensor of the TSL25X and TSL26X series [TAOS,02]. Aiming at different wavelength and amplification factors.

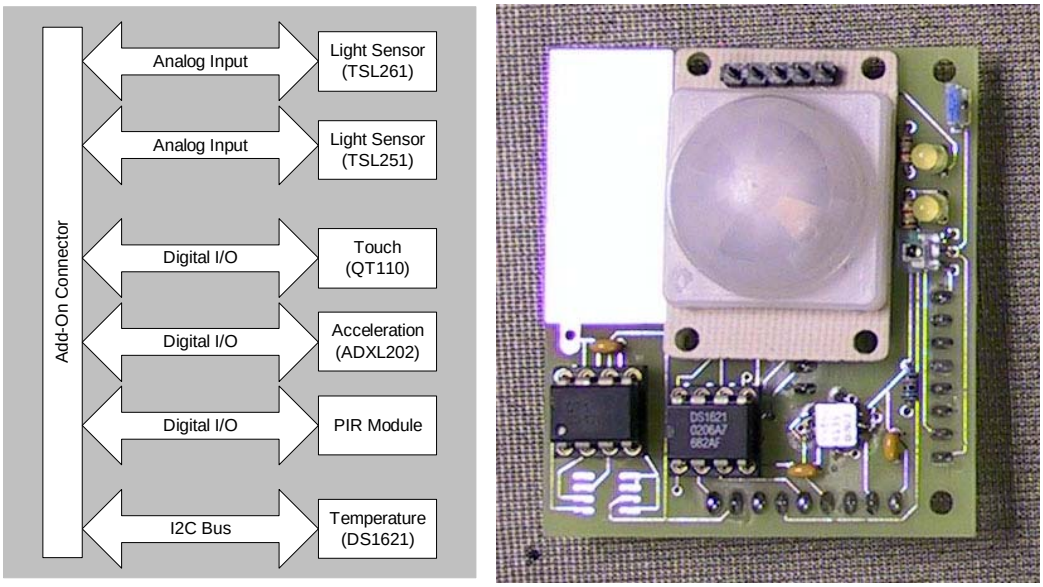


Figure 18: Sensor board block diagram (left). Completed sensor board (right).

- Acceleration/orientation in two dimensions. An ADXL202E sensor provides information about acceleration and that is dependent on the way the device is hold. If no acceleration occurs the output represents the spatial orientation.

- A capacitive touch sensor. To get information about user interaction a capacitive touch sensor is included. The sensing surface can be extended to any conductive surface on the artefact (e.g. a door handle can be made touch sensitive).

- A passive infrared sensor (PIR) module is mounted on the board. The digital output signals movement of people or animals around. The lens – a half sphere – offers a viewing angle of 360º with an opening of 180º. This sensor works only on the 5V version of the core board.

- Also a digital temperature sensor is on the board. The sensor is on the I2C bus and provides a resolution of 0.5º centigrade.

Beyond the sensors also two LEDs are included in the design. Having visual output on the boards makes it easy to give simple feedback, especially when debugging applications. The sensor board is depicted in Figure 18 (right). The full schematic is depicted in Appendix E.3: General Sensor Board Schematic.

To run the core board with a sensor Add-On as wireless sensing node, basic software is provided. It implements a sensing loop at the end of which the current results are sent via RF and also over serial line. The code is also a basic framework for more elaborate usage of the sensor Add-On.

A more advanced version of the sensing software is configurable. It also executes a sensing loop, but keeps a history and calculates basic statistics of the sensor values over time. The serial line can be used to change the configuration, in particular which sensors are read, the transmission format, the sensing speed, and how often a wireless transmission is carried out.

5.4.2.4 Load Sensor Board

To carry out load sensing experiments as describe in detail in section 4.4.2 and in [Schmidt,02] a custom hardware was developed. The result of the deployment phase resulted in a list of possible improvements:

- Increased AD resolution.
  To avoid the influence of single bit errors in the position calculation and also to increase the weight range of the objects that can be detected, an ADC with higher resolution has been chosen (an external 16 bit ADS8320 analog to digital converter). This chip offers only one input so a multiplexer was used.

- In order to improve the performance of the amplification instrumentation amplifiers (e.g. INA122 from TI) are used instead of Op-Amps. The INA122 offers an amplification range of up to a factor of 10000.

- As the load on the surface changes and as it is also desirable to be able to detect very light objects or minimal interaction, the factor for the amplification can be selected at runtime from the microcontroller. Two amplification factors are implemented (factor 150 and 1000) using two instrumentation amplifiers for each sensing input.

- Additional RAM.
  In some advanced scenarios it becomes eminent that keeping a history on the

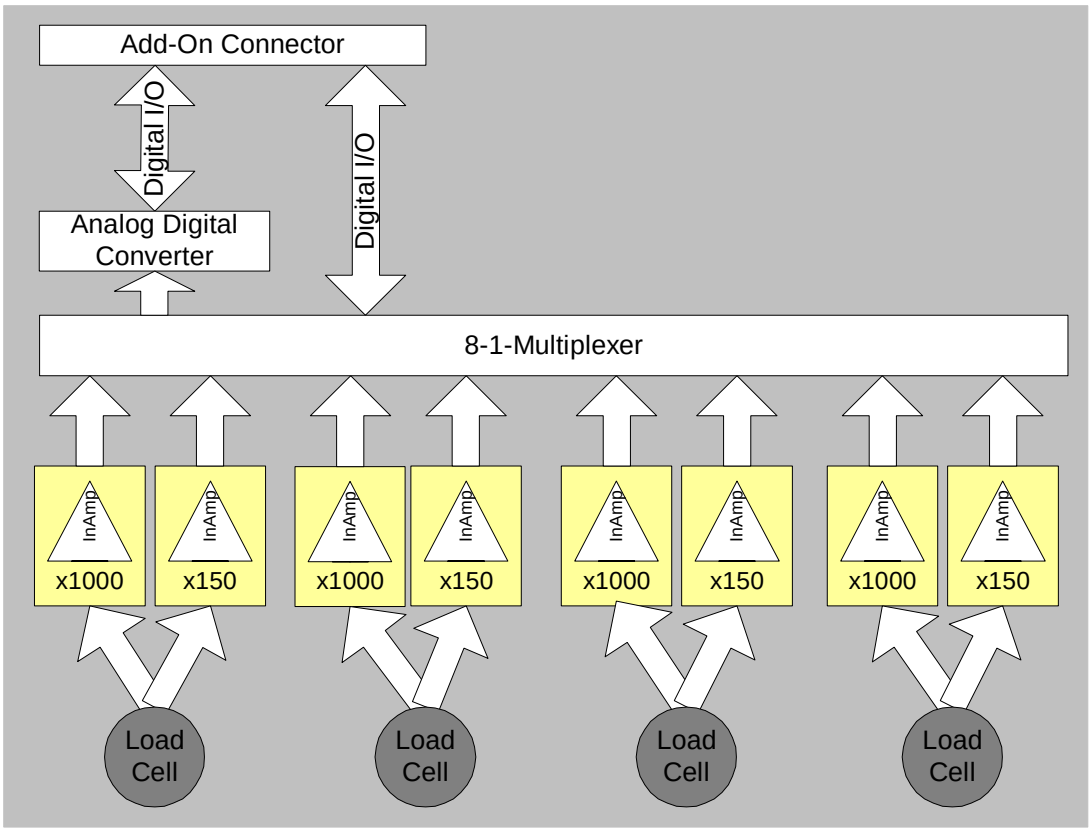


Figure 19: Load-sensing Add-On Board.

artefact is useful to detect more complex events or patterns of events. This requires additional storage, which is already included in the core board.

Instead of redesigning the original board the load sensing system is realised as an Add-On to the Smart-Its core board. The block diagram is depicted in Figure 19, and a photo is shown in Figure 20 (left). Technical details are included in Appendix E.4: Load Sensing Add-On Schematic.

The software framework is an extension of the software used in the experiments reported earlier. The software is configurable using a serial terminal which is connected to the serial port of the core board. In the simple version all forces are measured and the point of pressure is calculated. The sampling speed, communication interval, and the communication format can be selected. In the advanced version low basic weight events (object put at, object removed, etc.) are recognised and communicated. These programs are also the starting point for further developments using this Add-On board.

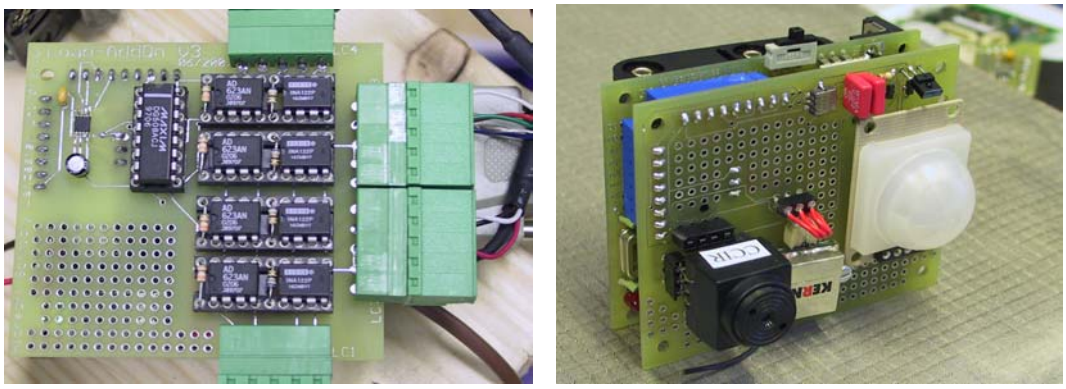


Figure 20: Load sensing (left) and wireless camera (right) Add-On boards.

5.4.2.5 Further Add-On Boards

Within different projects further Add-On boards have been developed and used. All boards have in common that they are build around the Add-On connector. The development is simplified because the core already offers digital I/O, analog inputs, and power. In particular the following boards have been built.

- Vision Sensor Board. This board is developed in the course of the Smart-Its project. It includes a CMOS-video camera, an analog video transmitter, several sensors (acceleration, light, PIR). Beyond the basic functionality of the core the software framework allows reading of sensors and controlling camera and transmitter, see Figure 20 (right).

- Weather & Health Board. The Add-On board is designed to include a number of sensors that can be used to monitor weather conditions. Additionally two further serial lines are included to connect a GPS unit and an Oximeter [Nonin,02]. The serial line of the core is connected to a mobile phone. Using this setup sharing of weather information can be realised. Additionally health monitoring and emergency calling is included.

- AM-RF Bridge. For certain applications – especially when physical size and power are a major concern and bandwidth is less important using AM transmitters is an option. The AM-RF Bridge is an Add-On that includes an AM-transmitter and receiver. Having artefacts that provided their information using an AM transmitter this module can be used to bridge it to serial line or to FM-RF.

- Experimentation Boards. Different types of experimentation boards have been developed. They contain solder holes for all pins on the Add-On connector and also a breadboard area. These boards make it relatively easy to test new hardware Add-On (e.g. a new sensor). In the evaluation workshop several new Add-On boards have been developed using these boards.

The complexity of designing an Add-On module is much less than of designing an entire system. In particular having modules, that can be separately debugged (hardware and software), speeds up the development process. As it will be discussed in more detail later (see chapter 8) people without a deep knowledge in electronics can add new sensors using this approach.

5.4.2.6 Communication & Backend Integration

The Smart-Its boards can be used in any of the system architectures shown earlier in this chapter. One option is to use a single board and connect this directly to a context consumer, e.g. the serial line of a PC or PDA. The other option is to use a single or multiple wireless boards and one core board as a base station. As the RF communication is realised as broadcast also multiple base stations can be set up, all receiving the same information.

The base station is the same hardware as the core board. It does not require any Add-On to be attached. In some cases it may be useful to attach a sensor board to the base station and use the unit twofold. Having the sensor information from the base station may be interesting when dealing with context proximity.

The basic software which is running on the core board acting as base station receives the information broadcasted by other units, checks that the packets are valid (e.g. the CRC is ok), and forwards them to the serial line. This code can be easily extended to filter packets or do further calculation on the incoming information.

On the backend systems (usually PCs and PDAs) the information has to be received from serial line. In different projects various solutions have been used. In particular implementations for Visual Basic under Windows, C/C++ under Windows and Linux, and JAVA under Windows and Linux have been realised. Additionally also a serial terminal is developed that can be used to communicate with the base station.

Depending on the application domain the data is directly used on the target device, or the information is handed on to the network, e.g. using the FuzzySpace (see next chapter for details), UDP broadcast, or offering the data via HTTP.

5.5 Discussion

In many application domains where context-aware applications are novel it is most often inevitable to build prototypes to assess the feasibility of a system. Building functional prototypes in size and weight appropriate to do user testing is usually not possible using standard hardware (e.g. PCs, PC104, PDAs). But the development of custom hardware systems is expensive and time consuming. Prototyping can be supported in various ways.

Two complementary approaches are:

- Easing the process of building full custom designed prototypes,
- Providing a flexible platform that can be used for such systems.

In the experiences gained in different projects it appears that these approaches are useful in different phases during the development process. In an early phase where the feasibility is assessed it is useful to have a quick solution to see whether or not it is useful to go on further and build a custom system. In this phase it is often acceptable if the physical constraints imposed by the application are not fully met. However even in this phase it appears to be problematic when the physical constraints are ignored. For example when assessing the feasibility of a context-aware system built into a shoe, having a sensing system mounted on a notebook computer does not provide a proof of concept.

In a latter phase of a project when experiments with real users are conducted (even if they are only users inside the living lab) it appears crucial to have devices that meet the actual physical constraints drawn from the host artefact. In the Mediacup [Beigl,02] project this was an important finding. In many cases building context-aware systems into everyday artefacts or devices – not changing the physical appearance significantly – will make the development of a custom hardware inevitable.

5.6 Summary and Conclusion

To prototype context-aware systems and in particular context acquisition systems more efficiently tools and mechanism for reuse are required. The design, development, and implementation include the overall system architecture, software, hardware, and communication. Therefore the problem becomes more general than in software development as all areas influence each other.

Three areas of work are described: libraries and building blocks, tools and methods, and a rapid prototyping platform for context acquisition systems. Libraries and building blocks are the way in which the experience in hardware and software development is accumulated and made accessible for further use. At the centre of building a development tool is a step-by-step method to develop context acquisition systems. The rapid prototyping system evolved from the developments in various projects. The components that are required in virtually any development are included in the core. Depending on the application scenarios and on the contexts of interest different Add-On modules providing specific sensor constellation can be added.

A detailed report on the evaluation of the prototyping platform is described in chapter 8. The often inherent distribution of sensing and perception has emerged to be a central issue, which is also assessed in the next chapter. Up to now the use of context and context-aware applications has been regarded only on the side. In chapter 7 these topics and in particular the impact on the human user and on the concept of human computer interaction is investigated in more detail.

Chapter 6

Distributing Context in an Ubiquitous Computing Environment

In a Ubiquitous Computing environment context is inherently distributed. Artefacts in the real world are physically distributed. Considering the original mean of context (‘context is what is surrounding something else’) and taking into account the model of aware artefacts it becomes apparent that distribution has a significant role. Distribution occurs on different levels, firstly on a conceptual level where information – that is regarded as context – is distributed, and secondly on an implementation level where system components, dealing with context are also distributed.

When implementing context-aware systems three main functional areas have to be considered: context acquisition, context synthesis, and context use. As artefacts in the real world are spatially distributed also the acquisition and use of context in Ubiquitous Computing is spatially distributed when bound to artefacts. For the synthesis or fusion of contexts no explicit locality is required, nevertheless the input to the process (e.g. various context sources) and the output are anchored in artefacts and therefore in space and hence this process is in general distributed.

Besides the spatial distribution of context there is also a distribution over time, because contexts are rather observed as states than as events and therefore time and in particular intervals play an important role.

A fundamental assumption is that the spatial and temporal relationship between context sources and context users is meaningful and of value for the development of applications.

As seen from the rational above and also when analysing implemented context-aware systems, it can be seen that distribution aspects are central to the realization of such systems. To ease the design of context aware systems in Ubiquitous Computing on a conceptual level a distribution model that incorporates the common issues is required. Furthermore to simplify the implementation and to support a rapid development process a mapping from the model to the implementation is desirable. Other concerns are support on system level (e.g. context middleware) and functionality offered by programming languages to make these distribution issues transparent for the developer.

Main goals when designing a platform to ease the development of context-aware applications in distributed settings are to enable:

- Easy sharing of context information that is created in an artefact with the other parts of the system.
- Seeing and having access to context information that is around an application.
- Spatial and temporal distribution of context information within the system according to human understanding of context but transparent to the developer.
- Offering a minimal and straightforward API.

Viewing context as what is around an artefact follows logically along the same lines as modelling context in a bottom-up approach. Context information that is created at a certain point in space and time – at an artefact – is visible and usable in a certain time interval and in a certain distance – in the vicinity of the artefact.

6.1 Human Understanding of Context

Before designing a platform for context distribution it is worthwhile to look closer at the human understanding and perception of environment, surrounding, and context. In particular spatial and temporal issues are of interest.

6.1.1 Spatial Issues

Humans are always at a certain position in space. Usually the current position – 'the here' – is the centre of action, perception, and attention. The perception, e.g. what someone feels, hears, and sees, is dependent of the position and the spatial distances to the source of context. This results in the fact that the context that is perceived is strongly dependent on the position where someone is.

Moving position is also a human way of selecting an appropriate context for the activity that is performed. An example is walking towards the lights when observing something very closely; the lighting condition – the context – is changed by changing the position. This is a very powerful concept and adapted in many location-aware applications.

Humans use space and locality as an efficient tool for structuring the environment and also to support tasks and actions [Kirsh,95], [Beigl,01]. Spatial arrangements of artefacts are a most natural way for humans to order things. These spatial arrangements play a vital role when interacting with objects. Especially the concept of co-location is powerful and very often used, e.g. the books that are physically close on the shelf are often also similar in content.

6.1.2 Temporal Issues

Similar to the 'here' in location humans are always at a certain point in time – the 'now'. This point of time is where one acts and also where one perceives the environment, analogous to the perception in space. The perception of contexts that are around is restricted to the point in time. The contexts to be seen at this moment in time must exist at the same time as they are encountered.

Unlike in space where one can decide where to go, it is not possible to make this decision in time. Humans can only interact with the environment at the very moment in time where they are. However mankind has developed strategies to extend the understanding and perception of time.

It is not possible to go back to another time, however by recording what is going on and replaying it, it becomes somehow possible to simulate a step back in time. A good memory or making notes are the obvious examples for this.

Stepping into the future appears more difficult as there is always some uncertainty about what is going to happen. However in life most of the actions rely on these everyday predictions of the future and humans have developed a fairly good understanding of how to make these predictions. As we influence the string of events in our life we also influence the future and therefore we are able to predict it to some extent. E.g. I am leaving home 25 minutes before I am going to meet someone at work to discuss a project. This example relies on various predictions, e.g. it will take me about 20 minutes to go to work, the other person I am going to meet will be there, the issues we will discuss are not yet solved, etc.. On a smaller time scale, e.g. seconds, humans are used to a very stable environment and therefore the predictions how our surrounding is going to look like in two second is close to what is observed now.

6.2 Properties and Principles of Context in a Distributed System

Perception of situations and acting in context is for humans a major part when interacting and communicating in everyday life. Taking into account that context is related to artefacts and also the observations how humans perceive context in space and time the following basic properties for context are stated. These basic qualities of context foster a systematic foundation for a system that supports nature-like context in a Ubiquitous Computing environment. The properties are extended by basic design criteria for complex and distributed systems.

6.2.1 Locality and Proximity

Situation and context can be seen as phenomenon that is related and bound to a particular place or region. The place or region where context information emerges – or that is assigned to this context information – plays an important role, especially in mobile and embedded systems. The place or region must not be seen isolated, it is always an attribute assigned to an identity, a process, a device, a task, an application, or to data. In mobile location-aware systems the position is an attribute of the device and implicitly of the user who is carrying the device.

Collecting data from the environment and acquiring context out of this data is inherently bound to a location. The readings are collected at a particular position and therefore they represent the context for this particular position or the area related to this position. The information is fully relevant at this position. Generally the relevance

of the data as well as the certainty on the correctness of the data declines with the distance from its point of origin. An example is measuring the temperature at a certain point. At this point the temperature is correct and relevant, however when interested in the temperature at a point nearby it is observable that with an increasing distance between the points the uncertainty whether or not the temperature is also valid for the new point gets larger. And when the distance between the point of origin and the point of interest is too large the reading is meaningless. This leads to the conclusion that if several sensors of the same type with similar quality exist the one that is closest to the point of interest is the most relevant to look at.

As seen from these observations locality of context is quite important and should therefore be included in the model as one of the basic principles. For the model the following aspects and requirements should be taken into account:

- context information has a point or region of origin
- at the point or region of origin the relevance of the context information is maximal
- the relevance decreases with an increasing distance from the point or region of origin
- if several sensors of the same type are available the one which is spatially closest has the highest relevance

6.2.2 Time

Time is for the human understanding and classification of situations a vital aspect. It is also highly relevant for sensor data that is acquired from the environment. Using concurrency or exploiting the fact that events take place coincidental or within close timely boundaries are a basic way of relating different aspects of complex situations.

Regarding the time aspects when acquiring sensor data and contexts a similar semantic as for location is observable. Values are created at a certain point in time; and these values are in general more relevant to an event that happens roughly at the same time than to an event that takes place much later or earlier.

The concept of time should also be included in the model, exploiting the following basic observations:

- context information has a time of origin
- the relevance is maximal at the time of origin
- relevance decreases with an increasing time distance from time of origin
- if several sensors of the same type are available the one which provides the most timely reading has the highest relevance

In certain application areas it may be useful and beneficial to relate the relevance to issues that are specific to the application rather than to the temporal and spatial distance. In the model and platform described in this chapter this case will not be further regarded, as for many application areas in Ubiquitous Computing time and space are a prime concern.

6.2.3 Independence Between Acquisition and Use

The context, regarded as the type of situation that surrounds something else, is widely independent of the way it is used. From everyday experience we know that the perception of a situation does not change the current situation. Similarly in a context-aware system, acquisition of context does not influence the context. Again as we can recall from everyday experience after perceiving a situation the action taken may have an influence on the future situation. This again is similar in a context-aware system, knowing a context and changing the behaviour of an application according to the current context may change future contexts.

For modelling and designing a distributed platform it is desirable to reduce the number of dependencies as much as possible. To realise a loosely coupled system identifying issues that can be dealt with separately is of major importance.

There is a great variety of methods and technologies available for the acquisition of context in Ubiquitous Computing environments. The process of inferring context from data collected in the environment is itself usually a multi-level process that is conducted by components that are independent to some extent [Schmidt,99c],

[Golding,99], e.g. sensors, feature extraction, and perception of context should be independent.

It is also desirable that the algorithms and methods that supply context do this in a most general way abstracting from supplying context to a specific application. If context is delivered in a general way and independent of a specific application it becomes also feasible to develop context-aware applications without a particular sensing environment in mind. In the area of location aware systems this is already the case. Everyone can develop an application making use of location, e.g. by using a general description such as geographic coordinates, without knowing what the actual sensing system will be.

Having independence between context acquisition and context use also makes it possible to simulate either side to test and debug the other.

This compiles into the following wish list summarising the requirements:

- context acquisition and context use is highly independent
- all levels in context acquisition are highly independent
- it is possible that more entities that supply context exist independently (even for the same context)
- it is possible that more entities that uses context exist independently (even for the same context)
- applications are modelled independent of time and location but using these properties implicitly

The property of independence between context acquisition and context use is not necessarily general. In some application areas where acquisition is specifically designed for an application it may advantageous not to insist on independence. However this has usually the price of less flexibility and greater complexity.

6.2.4 Distribution and Scalability

In everyday life humans have a great ability to filter information and shift to information that is relevant, one prominent example is the so called 'cocktail party effect' [Handel,89]. This basic mechanism protects our information processing system from information overload, because the processing system is limited. Similarly the question of scalability arises in systems that deal with many stimuli from the environment and potentially a vast number of contexts.

As artefacts and infrastructures are inherently distributed it is one obvious approach to exploit the spatial and temporal properties of context to achieve scalability. This follows the concepts described earlier on the basic properties of locality and time for context. Scalability over time can be realised by modelling data in a way that it disappears after a certain time.

The concept of spatial and temporal scalability can be illustrated by the following examples describing human perception. The perception of sound scales spatially, as with an increasing distance to the source of sound the volume as well as the quality decreases. From a certain distance the sound not audible at all. The scalability over time can be illustrated when considering the human perception of smells. At the moment they are created the smell is at the maximum (e.g. while cooking). Over time the smell fades till it is not recognizable anymore. If this scalability would not be 'built-in' to our world life as we know it would be hardly possible; just imaging sound would not be locally limited.

The following aspects related to scalability and distribution are taken into account for building the model:

- the distribution of information is locally restricted – localised scalability
- the lifetime of information is restricted – scalability over time. (In the case where is no new information added the amount of data decreased over time.)
- the spatial distribution of individual components is a basic property

6.2.5 Transparency

Again looking at the human way of perceiving context it becomes apparent that the spatial and temporal distribution is transparent. It happens to be around without further considerations. The reference to a person's position and to the current time is implicit and usually unnoticed.

When designing a system that supports the use of context in applications it seams desirable to offer a similar degree of transparency about the distribution of context. The challenge is to create a system that provides context information for applications dependent on the temporal-spatial relationship between the application and its environment. The underlying mechanism for distribution and spreading of context should however be transparent for the context user as well as for the one who produces context information. The components that offer context should be able to influence the spreading of the context they are creating.

In summary this results in the following demands for the model and architecture:

- context is always bound to the current location and time
- distribution and fading mechanisms are built into the model and the platform
- context is transparently distributed for context creators and users

The description of the nature of context in a Ubiquitous Computing environment – seen from a human perspective – led to the extraction of basic properties. These observations become the foundation for the distribution model and platform described in the reminder of this chapter.

6.3 Describing and Accessing Context

Context is a description of a situation, as defined in chapter 3. To make use of context in a distributed system and also to make an implementation feasible a specific data structure has to be defined. Furthermore there is also need to specify ways by which contexts can be compared, because that is a prerequisite for accessing context information.

6.3.1 Describing Context

As seen in chapter 2 context can have many different structures and forms of appearance. Providing a single formal model to suit all the needs of context-aware applications appears similar to the problem of providing a single formal description for all world knowledge. To avoid the task to create a comprehensive world model (and thereby avoiding the risk of creating a model that explodes in size and complexity) it is decided that the distribution architecture does not enforce a particular way in which context is described.

The context data structure is used to communicate information about the environment between different components in the system. In this model no implications are made on the actual content of the data structure, there is no fundamental difference between a sensor value, a calculated feature, and an inferred context in the data structure. There is no basic restriction to the domain or to the values themselves; they may be scalar, vectors, or arbitrary character stings. The choice of format for the description of context is of minor relevance for distribution and communication, however for building applications it becomes more important.

As suggested from the observations above contexts have a reference to a location and also to a point in time. This reference is however implicit given by the position of the artefact or the users and therefore there is no need on the level of a context to include this information. As location and time information is relevant for implementing distribution they are modelled in distribution platform.

Context information is represented as a 3-tuple:

```
C = (ID, context-data, probability)
```

The context information (*C*) is identified by an *ID* and contains data in *context-data* which is a data type that is defined by the system developer. In the structure given above the provision of probabilities for the context value is included. If the perception system is offering this information it can be included.

The information about place and time are not contexts on their own. They are treated as meta-information about a context. This information is used in the model as well as

in the platform to control the spreading of contexts. Time and place are always related to an entity such as a person, an object, and an application. Even if only location is used as context this is implicitly assigned to an entity.

6.3.2 Content-Based Access to Context

The model proposed provides content-based access to context. To realise content-based access context-templates are used. These templates are a mechanism to describe conditions that allow filtering for contexts that have certain properties.

Such a template is tailored to the data structure. By creating a context template basically a search query over all existing and visible context information is specified. Semantically all conditions given in the template are AND-connected. Only contexts that match all conditions specified in all categories of the template are selected. A wildcard symbol (‘*’) is provided that matches anything. By these means more general context templates can be specified.

A context template consists of three parts:

```
CT = (ID|*, context-data template|*, minimal
probability|*)
```

If in the template an *ID* is given then only context elements that have exactly this ID are selected. The ID is usually related to a specific domain, typically to a specific type of context supplier. The context-data template gives a description which contexts should be matched. The description template has to be tailored to the description format used in the context element. In the simplest case, when the description is just text, the template holds single words and for matching a sub-string comparison is performed. When providing a minimal probability this is included when matching, too. Only context that have at least this probability assigned are matched. An example of a more structured context representation and template is published in [Schmidt,01].

6.4 Modelling the Distribution of Context

By creating a model for the distribution of context in a Ubiquitous Computing environment many of the ideas and concepts discussed earlier in this chapter are put together in one coherent body.

A basic assumption of the model, and also requirement to make use of an implementation of the model, is that context information comes into existence at a well-defined position and at a well defined time. The same is assumed for components that use context; they are always associated with a point in space and time.

The model describes the spatial and temporal distribution of context with relation to the point of origin of this context. The basic concept of the model is that context information is associated with a relevance that is decreasing with the spatial and temporal distance from the origin.

6.4.1 Fuzzy Sets

As known from everyday experience situations change most often gradually. E.g. there is no sharp cut about how far you can see or hear something. For contexts it seams not appropriate to have borders where they are true on one side but not true on the other side. Here also a notion of fading is required in the model.

This fading, or fuzziness, is related to the relevance of the context. It is modelled similar to the idea of fuzzy sets and fuzzy logic [Zadeh,73], [Traeger,94]. In fuzzy sets the basic idea is that the membership of an element to a set is not just binary. It is rather fuzzy – meaning that an element has a degree of membership to a set. A single element may also belong to a number of different sets, having for each a degree of membership assigned. Typically the degree of the fuzzy membership is modelled by using functions that give a relevance value between 0 and 1, typical triangular and trapezoidal membership functions are used.

To model the distribution of context this concept of relevance is used for representing the spatial and temporal relevance of contexts, based on the distance from the point of origin. Using this method with appropriate functions it is possible to assign each context value a temporal and spatial relevance at any given point in time and at any location in the system.

6.4.2 Relevance Based on Time Difference

As each context value is created at a certain time and this time is known, it becomes possible to assign a relevance to the context information at any point in time based on the time distance between the two events. To model the relevance value any arbitrary

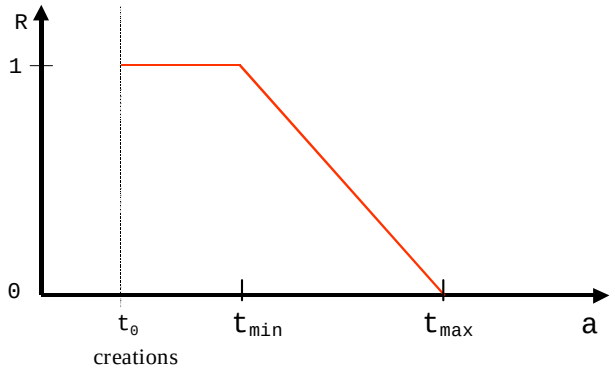


Figure 21: Example of a relevance function. The temporal relevance of the context value created at t0 decreased over time.

monotonous decreasing function can be deployed to describe how the relevance changes with an increasing time distance between creation of the context and the time when it is of interest to an application.

Triangular and trapezoidal functions are used, similar to membership functions in fuzzy logic systems. In Figure 21 an example of a decreasing relevance over time is given. In this example the context value was created at *t0* and the assumed function is a trapezoidal function.

In principle any function could be used to calculate the temporal relevance. As seen in Figure 21 the function translates a time difference into a relevance value between 0 and 1. To keep the semantic of the model coherent with the everyday experience as described earlier it is required that the function is monotonously decreasing. The relevance gets smaller the longer the creating of the contexts is ago. In certain cases it may be useful to consider a negative time difference. This is the case if the creating data of a context that is in the future, but the context value is already around. Especially when contexts can be predicted, e.g. someone is driving onto the parking then it can be predicted that the context 'person entering the building' is going to be created in the near future. In these cases a function that is symmetric around the creation time may be used.

The relevance function is a mapping from the domain of the real numbers (representing the distance in time) to the interval [0,1], which is representing the relevance:

$$f_{RT} : R \rightarrow [0,1]$$

Further requirements for the relevance function that have evolved from observations stated earlier in this chapter, in particular that the relevance becomes smaller with a longer time that has passed.

$$\Delta t = t - t_0$$
$$|\Delta t_1| \geq |\Delta t_2| \Rightarrow f_{RT}(\Delta t_1) \leq f_{RT}(\Delta t_2)$$

Additionally it is required that the relevance becomes zero when the time distance between the creation of the context and the current time is greater than a certain value:

$$\exists t \in R : t < \infty \wedge f_{RT}(t) = 0$$

The following trapezoidal function fulfils the requirements stated. The parameter t_{max} defines the maximal temporal difference from which on the relevance is zero. The parameter t_{min} is the distance at which the relevance is still one.

$$f_{Trapezium}(x) = \begin{cases} 0 & : x \geq t_{\max} \\ 1 & : x \leq t_{\min} \\ \dfrac{t_{\max} - x}{t_{\max} - t_{\min}} & : \ else \end{cases}$$

This relevance function is depicted in Figure 21.

6.4.3 Relevance Based on Distance

As each context value comes into existence at its point of origin, the model describes the spreading of the information from this position. The main purpose is to provide information on the relevance of the context at any point in the system. The spatial relevance is defined similarly to the temporal relevance using functions inspired from a fuzzy system. In Figure 22 relevance of a context in a 2 dimensional location model is illustrated.

As foundation for the spatial distribution model any location model that supports the calculation of a distance between two points may be deployed. Location models can be built on coordinates, where the distance calculation is simply based on an arithmetic model. However the model is not limited to that, symbolic location systems that provide a functionality to calculate distances can also be used. These models are of particular interest when dealing with human readable location identifiers, such as room numbers, levels, building, streets, towns, and countries. In these cases the result of the distance calculation may be discrete. A more detailed discussion of location modelling can be found in [Leonhardt,98]. In the reminder of the section two dimensional Cartesian coordinates are assumed, which eases the visualization of the relevance function.

In Figure 22 the spatial spreading of a context element with a point of origin at x=650 and y=400 is visualised. The function used is a trapezoidal function where the argument is the distance between the point of origin of the context value and any other point. So it is possible to calculate for any point in the x,y-plane the assigned relevance of the context, displayed in the z-direction. As seen in the figure the relevance of a context is maximal at the point of origin, decreasing with a greater distance, and from a certain distance on the relevance becomes null. The following Euclidean Distance function in a two dimensional coordinates system is used:

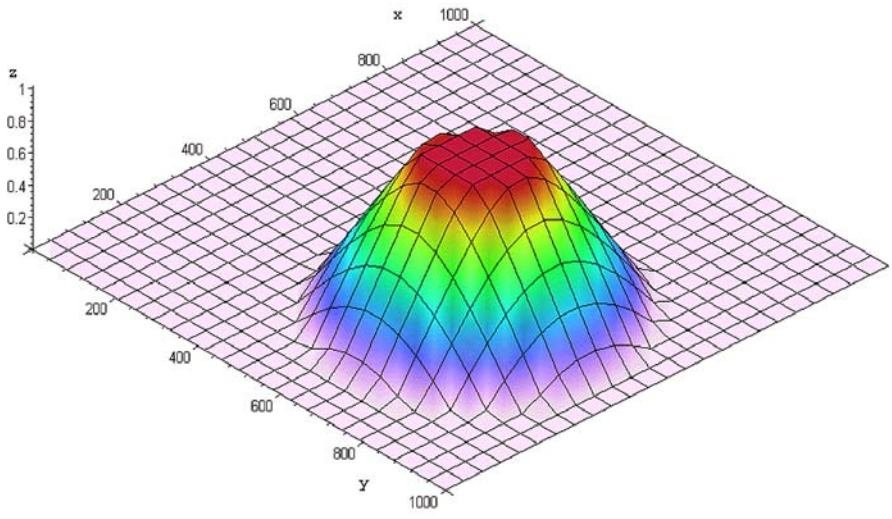


Figure 22: Visualization of spatial relevance.

$$\Delta l((x_1, y_1),(x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

For the selection of a spatial relevance function the same requirements apply as for temporal relevance functions discussed above.

6.4.4 Transparency

Spatial and temporal relevance become inherent properties of context anchored in the distribution model. However for the context producer and also for the context consumer these properties are transparent at the first place.

Context producers create contexts without caring about distribution and context consumers. The users of contexts take the context information that is visible, without caring who supplied them. To make this transparency feasible, distribution and spreading as well as a mechanism to filter context information has to be built into the system. In the system described here this is provided by assuming an information pool on to of which temporal and spatial relevance is defined as modelled above.

6.4.5 Requirements

The model introduced here assumes that specific information about the creators and users of context is available in the system.

In particular it is required that for each component that creates context and also for each component that makes use of context the physical location is know. The model does not impose restrictions on how exact the location has to be known. Depending on the applications the location information required may be the room where a component is in, the building, or just the site. In many practical cases this information is available without having an explicit location system. For components that are embedded in the environment (e.g. sensing system) the location can be hard-wired at installation time, for mobile components this coarse location information may be deduced from the communication system used. For applications where a fine grained location is required an additional position system may be used.

Furthermore communication is also required between all components in the system. The communication system must provide basic functions that allow components to

communicate with other components in the system without prior knowledge of these. A typical example of such a communication primitive is a broadcast mechanism.

To calculate the temporal relevance time information is required, too. As communication is available time may be acquired using communication or time stamps may be added while communicating to avoid having distributed synchronised clocks.

6.5 FuzzySpace – A Distributed Communication Platform

The model to spread context information and the change of relevance based on temporal and spatial distance is created with regard to context. However it can be observed that for other communication purposes in Ubiquitous Computing the notion of relevance and spatial and temporal distance is also applicable.

In this section a communication platform incorporating these features will be presented. The FuzzySpace is a general communication platform with an underlying semantic distribution model targeted for Ubiquitous Computing environments. In a later section the FuzzySpace will be refined for the use in context-aware systems.

6.5.1 Architecture

This communication architecture consists of three types of components: message producers, message consumers, and the FuzzySpace as communication platform in between. The FuzzySpace is based on a tuple space that is extended with spatial and temporal semantic. The general architecture is depicted in Figure 23.

Components that supply messages to the system are denoted as message producer, components that read messages from the system are denoted as message consumer. Each message producer and consumer is at any given point in time at a specified position.

Consumer and producer can communicate with the FuzzySpace. If a component supplies messages and also reads messages it incorporates a consumer and a producer.

6.5.2 FuzzySpace

The FuzzySpace realises the underlying communication platform. It is developed as an extension to tuple spaces [Gelernter,82]. The FuzzySpace offers the possibility for independent communication between components without prior knowledge of each other. To provide a communication semantic similar to the observations stated earlier a spatial and temporal spreading semantic is built into the platform. Designing FuzzySpaces as an extension of tuple spaces was a deliberate choice as it allowed to reuse findings and implementations available.

All elements in the FuzzySpace are inherently bound to a location and a time. When communication facilitates the FuzzySpace each element has assigned a relevance value dependent on the time and location of access.

In tuple spaces that are deployed for loosely coupled systems common basic operators are *add*, *read*, and *remove* [Wyckoff,98]. The FuzzySpace offers a subset of these operations, which are extended by location and time. Elements that are added to the FuzzySpace are always associated with a location and are implicitly associated with a time – the time when they are added. When reading from the FuzzySpace this is similar – it implicitly happens always at a certain time and is explicitly bound to a location. Typically the location is related to the device that is communicating.

6.5.2.1 Operators

The following operators are derived from standard operators for tuple spaces extended to support the requirements stated above. The main differences result from mechanisms to dynamically include temporal and spatial relevance with each tuple. The primitives presented here are basic operators for the management of FuzzySpaces

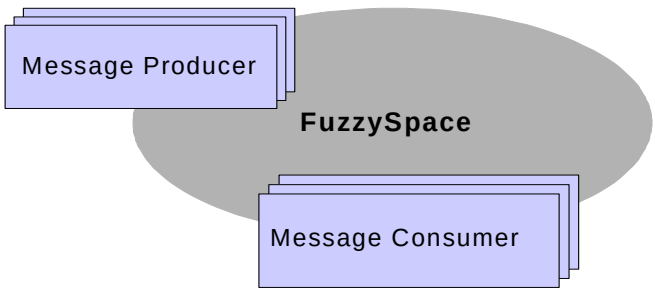


Figure 23: components of a distributed communication platform.

as communication instances as well as primitives that allow system components to access the FuzzySpace.

The following basic operators support the creation, deletion, and querying of particular instances of a FuzzySpace.

```
CREATE_FS(<description>?, <FuzzySpace_Handle>!)
```

The function `CREATE_FS` creates a FuzzySpace in the system. The only input parameter is the description for the instance of the FuzzySpace. The description is an arbitrary data structure. In the implementation a string of characters is used to represent the description, this can be used to provide a free form description or to store a structured description by using data types on top of the character string, such as an XML description. The description can then be used to identify FuzzySpace. The return value is an object that is the identifier for the FuzzySpace.

```
DROP_FS(<FuzzySpace_Handle>?, <Status>!)
```

Using the function `DROP_FS` a FuzzySpace can be deleted from the system. The input parameter `<FuzzySpace_Handle>` identifies uniquely the instance of the FuzzySpace that should be deleted. When an instance of a FuzzySpace is removed from the system all data that is still in the FuzzySpace at this time is delete as well. Ongoing communications between other components in the system and the FuzzySpace are terminated, too. The return parameter `<Status>` offers information whether or not the operation was successful.

```
DESCRIBE_FS(<FuzzySpace_Handle>?,<description>!)
```

The function `DESCRIBE_FS` is used to access the `<description>` for a FuzzySpace that is identified by the `<FuzzySpace_Handle>`. The description that is retrieved with this command was stored with the create command.

```
QUERY_FS([<search_term>?],{<FuzzySpace_Handle>}!)
```

By using the function `QUERY_FS` the handles to all available FuzzySpaces can be retrieved from the system. The function returns a set of handles that identify

FuzzySpaces matching the query. When no <search_term> is supplied all handles available in the system are provided as a result. If a value for this optional parameter is specified, only FuzzySpaces that contain the search term in their description are returned.

System components that use the FuzzySpace to communicate, such as message producer and message consumer, need basic operators to provide and read data. In contrast to standard tuple spaces no operator for removing data elements is required. The existence of elements is strictly defined by the characteristic of the spatial and temporal distribution. Operations are executed at a specific position and at a defined point in time, these values are represented by the variables <Location> and <Time> which are used in the primitives.

```
ADD(<FuzzySpace_Handle>?, <Element>?,
<Location>?, <Time>?,
<spatial_Relevance_Function>?,
<temporal_Relevance_Function>?
<Status>!)
```

The function ADD is used to add elements to an instance of a FuzzySpace that is identified by <FuzzySpace_Handle>.

The <Element> can be any vector or scalar value. The time and spatial position where an element is added to the FuzzySpace is encoded in the parameters <Location> and <Time>. Further two parameters are used to specify the functions that describe the spatial and temporal relevance of the element that is added. The return parameter <Status> provided information whether or not the operation was successful.

```
MATCH (<FuzzySpace_Handle>?,
<Element_template>?,
<Location>?, <Time>?,
<Minimale_spatial_Relevance>?,
<Minimale_temporal_Relevance>?,
{<Element>}!)
```

The function `MATCH` realises a query for elements in the fuzzySpace identified by the handle. The query is related to the position and time encoded in `<Location>` and `<Time>`. All elements that have a spatial and temporal relevance which is at least as specified (`<Minimale_spatial_Relevance>` and `<Minimale_Temporal_Relevance>`) and match the template provided (`<Element_template>`) are selected. These elements are returned in a set together with their relevance values.

6.5.2.2 Message Producer

Message producers are components that generate messages and add them into the FuzzySpace. The message producer utilises the ADD-operator to put elements in the FuzzySpace.

The parameters for using the ADD-operator are gained as follows. The handle to the FuzzySpace is either predetermined (in the case where systems only use one FuzzySpace) or determined using the QUERY_FS-operator. The element that is added is the message itself. The location and time are implicitly given by the time the ADD-operator is executed and by the whereabouts of the component at this time. When implementing a message producer there is knowledge about the semantics of this component, especially how far these messages should be visible and how long they should be available. This knowledge about the specific instance of the message producer is encoded into the spatial relevance function and into the temporal relevance function.

A message producer is characterised by the following 4 parameters:

```
MP=(<current_location>, <current_time>,
<spatial_relevance_function>,
<temporal_relevance_function>);
```

6.5.2.3 Message Consumer

Message consumers are components that request messages from the FuzzySpace. The message consumer utilises the MATCH-operator to ask for elements that are of interest and that are available in the FuzzySpace.

Similar to the message producer the handle to the FuzzySpace is either predetermined or gathered via the QUERY_FS-operator. The consumer uses an element template to filter results from the FuzzySpace. This template is dependent on the data structures used. The time and location parameter are filled with the current time and current location of the component. A further mechanism to filter the result set is to state minimal relevance required. Only elements that have a higher local and temporal relevance are then matched by the operator.

A message consumer is characterised by the following 4 parameters:

```
MP=(<current_location>, <current_time>,
<required_spatial_relevance >,
<required_temporal_relevance>)
```

The result can be used by the message consumer directly; as the result set is ordered one option for the message consumer is to use only the first element that is returned and not further process other elements.

6.6 A Distributed Context Platform based on FuzzySpace

The FuzzySpace platform is designed as a general communication platform for Ubiquitous Computing environments. Many of the design decisions however relate to the notion of context in such systems, therefore it is obvious to utilise the FuzzySpace as a platform for distributed use of context.

6.6.1 Architecture

The architecture is a more specialised form of the general FuzzySpace. In Figure 24 the architecture is depicted, comprising of a FuzzySpace as communication space, a context producer, a context consumer, and a context abstractor.

Applications and devices in such an environment can embody multiple components. The minimal requirement for an application that uses context is to be a context consumer. A device or application that generates context must include a context producer. Applications that create new contexts based on already existing ones are incorporating a context abstractor.

The FuzzySpace has to be available at least once in the communication range of all components participating in the system. Using a single instance results in a centralized system, running several instances within different components results in a distributed setting.

6.6.2 Context Supplier

A context supplier is a component that provides context to the system. In the architecture presented here, context suppliers do not need to have someone who is interested in the context they supply. A context supplier can be a simple sensor that communicates raw values or a complex recognition system providing high-level context information.

The contexts that are provided to the system are based on the data structure agreed in the system. The context supplier is uniquely identified and provides contexts from a certain domain. The context information created has assigned a time and location of origin. This is usually the physical or logical place where the context supplier is located. As the context supplier is built as a specific recognition system, domain knowledge is available during implementation. This domain knowledge is then used to assign the context value with describing features. This knowledge is also used to define the temporal and spatial distribution rules.

A context supplier has a number of static properties that are defined at the design time of the context supplier.

```
CSs = (ID, description
spatial_relevance_function, temporal_relevance_function)
```

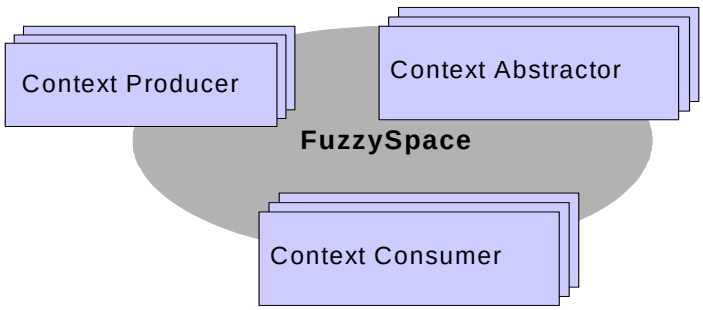


Figure 24: components of a distributed context platform.

These properties are valid over the lifetime of the context supplier. The properties *ID*, and *description* are related to the context data structure. The spatial and temporal relevance functions define the spreading characteristics of the context that is produced within the FuzzySpace. The model is open about the functions that can be used provided the function has the properties explained above.

For each value that is created and supplied there are also dynamic pieces of information in the context supplier:

```
CSd = (ID, context-data, probability, location, time)
```

The *ID* is not dynamic, but necessary to mach it with the static part of a context supplier. The *context-data* is the actual value of the context that is supplied. This value is meaningful for the place and time given by the parameters *location* and *time*. If the context supplier can determine the probability of the value this is also provided.

The static parameters (CSs) are defined when a context supplier is instantiated. The dynamic parameters (CSd) are gathered when a value should be provided to the system. At this point the information is used to complete the data structure that is needed to call the ADD-operator. In some case it may be useful to update the static parameters, in particular when the spreading characteristics change, during runtime.

6.6.3 Context consumer

Context consumers are components that use context information available in the system. A context consumer has access to any context element in the FuzzySpace that is visible to the component. A context element is visible when is has a temporal and spatial relevance related to the context consumers position and request time which is greater than zero. As a prerequisite to determine the local relevance it is necessary that a context consumer knows its current position. Context elements in the system for which the relevance is zero with respect to a context consumer are not visible for the component. In general context consumers are built to perform a certain task (e.g. a context-aware application). The domain knowledge for the application determines the contexts that are of interest to this particular context consumer.

Working with contexts as facts that are around, as motivated in the introduction to this chapter is realised in this way. Context is information that is around – whether or not we need it. For the context consumer component – or in particular for the developer of such components, it becomes transparent where and how the context is created and how it is spreading.

A context consumer is a dynamic structure, comprising the following properties:

```
CC = (location, time, element template,
minimal local relevance, minimal temporal relevance)
```

The values for *location* and *time* are determined by the current logical or physical position of the component. The *element template* is used to specify in what contexts the consumer is interested. This template is either static or dynamic – depending on the application. If the template is static it is specified when the component is initialised. In the case of a dynamic template it can be changes at any time during runtime. The minimal values for local and temporal relevance are also regarded as dynamic parameters. For certain applications however they may be specified during instantiation and not changed afterwards.

The properties of the context consumer are used to fill the parameters required to call the MATCH-operator to access the contexts:

```
MATCH (<FuzzySpace_Handle>?, element template, location,
minimal local relevance, minimal temporal relevance,
{<Element>}!)
```

The result received by the context consumer after the request is a set of context elements that meet the requirements specified in the request. The set can be empty; it can have exactly one element, or more elements. The usage of the context information is encapsulated in the context consumer component.

In the general case a context consumer is a context-aware application. The usage of context is completely separated from the acquisition of context. This allows writing and specifying the application without a specific infrastructure in mind. Therefore

context suppliers and context consumers can be developed and tested independently of each other.

6.6.4 Context Abstractor

A context abstractor is a component that reads context information from the FuzzySpace and also puts new context elements into the FuzzySpace. A context abstractor has the properties of a context consumer as well as of a context producer. The basic concept of a context abstractor is to take context that is available in the system and to generate new contexts and to provide these again. The functionality is similar to the basic principles known from black-board architectures.

Typically this can be regarded as a step of context abstraction, where simple context information (e.g. sensor values) is combined to more abstract context information that is then provided again. Context abstractors can also operate on the relevance functions associated with context elements in the FuzzySpace.

The location of a context abstractor is usually logical, because it is a process that is running somewhere in the network. However it is important that there is a location assigned to each subcomponent – the consumer and the supplier. These logical locations do not need to be the same. This opens means to transport context information from one physical location to another one.

6.7 A Context Library

To access the distribution platform a context library is implemented which provides the basic functions in a simple application programmer interface.

Access to all context information is implemented using this library. As introduced earlier a major goal is to make it as easy as possible for the application programmer to use context – transparently of distribution and acquisition issues. When designing the context library, a main issue was to keep the number of functions and their complexity minimal to make it usable without extensive training. The functions offered in the library are designed to support the implementation of software that consumes context as well as for software that produces context. The following four functions are the essential core of the library:

- **`InitContextLib()`**

  before functions in the library can be used the library and the underlying system must be initialised. This is done by calling this function. In particular network connections are initialised in this stage.

- **`InitContextPos(<pos>)`**

  As the physical position is a key property in the system the location information must be provided. The position can be altered at any time. All further access functions assume the last given position as valid.

- **`GetContext(<name>|<id>|*, <element template>|*, <min_local_relevance>|*, <min_temp_relevane>|*, <P>|*)`**

  From the application the current value for a context can be requested at any time. This is done using a context template. The parameters of this function are used to create a context template that is used in the MATCH-operator. Is in the system a value available that matches the template then this is returned by the function. To compose templates, wildcards (‘*’) can be used instead of a parameter value. Additionally also minimal values for the local and temporal relevance as well as for the probability associated with a context can be specified. The result is a character string that contains a formatted list of all matching context values.

- **`PostContext(<name>|<id>, <element>|<URN>, <P>)`**

  For applications that provided context information, such as context producers or context abstractors this function offers a simple mechanism to add context information into the FuzzySpace. The application identifies the context element by a name or identifier, supplies a context value or a reference (URL) to it, and the probability for the context value.

The implementation of the library and in particular the data types used in the function are dependent on the class of device (MCU vs. PC), programming language used and also the underlying operating system. In particular the return values are designed to fit the requirements of the platform. Within various projects, implementations have been

assessed for different platforms, in particular C and JAVA on PCs and C on a microcontroller. In the first cases Ethernet was used as communication media on the microcontroller serial line and low bandwidth RF was used.

6.8 Discussion

The distribution model and platform is closely tied to the concept of time and space. Time and space are obvious restrictions when dealing with subjects and artefacts in the real world. However one major advantage of computer systems is that restriction on time (data storage) and limitations in space (networking) can be overcome. Using networks and storage allows that information can be made available always and everywhere. Nevertheless the limitations introduced in the model were done so deliberately. From the experience gathered from different projects restricting the visibility of information – and in particular context information – in time and space makes it easier for developers to use. By these means there is no need to subscribe or register to get a specific context, this happens implicitly by being at a certain place at a certain time which simplifies the API. The model presented here shows an example of how information distribution is strictly modelled on time and space. In cases where the application domain imposes different requirements, the distribution rules may be modelled differently.

Running all communication between context providers and context consumers via a distribution platform may decrease the performance of the system. However using a dedicated communication platform, debugging and simulation becomes much easier. All messages that were sent can be traced and this allows to find components with malfunctions. Furthermore the decoupling of components makes it easy test context producers and consumers independently. A new context provider can be tested and the output received by the FuzzySpace can be used to ensure that the component works as expected. Also it is possible to run context consumers (e.g. context-aware applications) without having a sensing infrastructure in place. The sensor infrastructure can be simulated using software that produces context information. In different projects programs using graphical user interfaces have been developed where the context values could be selected. The component behaved then as a sensing system providing the information to the FuzzySpace.

Restricting information to a particular location and time can provide a mechanism to realise access control. Only components that are in a particular location at a specific time can see the information. This model takes then the security from the computer system level into the physical domain. This is however not without problems. To make such a system safe it would require mechanisms that ensure that time and location are authentic and that these parameters can not be faked by context consumers and producers. Furthermore it is necessary to ensure that all components running in the system are trusted and do not violate the distribution rules. If for example a component keeps information that was seen at a certain time in its storage and uses this information later, then it is not possible by the system to ensure that the data is not visible after a certain time. Similarly if a component in the system has access to the network and sends data received at a certain location to some other physical location the spatial distribution can not be assured.

6.9 Summary and Conclusion

Context is not an abstract concept; it is a part of everyday life. Situations happen timely and locally. Human understanding of situation and context is always influenced by these basic properties. These observations and additional design principles of distributed systems constitute the foundation for a distribution model. Motivated by the fact that changes in the real world are usually not abrupt, concepts from fuzzy sets are also included in the model.

These ideas combined lead to the FuzzySpace distribution model and platform. The FuzzySpace is a tuple space that is extended by a spatial and temporal distribution semantic. Elements are entered into the FuzzySpace at a location and time. Based on distance measures a spatial and temporal relevance can be assigned to each element when this is accessed at a particular location and time. Operators to add elements and to retrieve information provide the interface to the FuzzySpace.

Based on this distribution platform message providers and message consumers can communicate. Sharing and using context information is modelled on top of this general Ubiquitous Computing communication platform. Context elements are added by context providers, can be retrieved by context consumers, and context abstractors

can generate new context information based on information already available combined with domain knowledge.

The FuzzySpace is an example of a communication platform that is designed to cater for specific needs in Ubiquitous Computing environment. As such environments and also the usage scenarios differ greatly it is unlikely that one platform fits all needs. Nevertheless the example given in this chapter shows a concept that is generally applicable: modelling system properties into the communication platform.

The mechanisms presented in this chapter aim at easing the development of distributed context-aware systems. In the next chapter the implications of using context on the user interface are investigated.

Chapter 7

Interactive Context-Aware Systems

The availability of context and using context in interactive applications offers new possibilities to tailor applications and systems “on-the-fly” to the current situation. To show how context influences and often fundamentally changes interactive systems first a brief introduction to interactive applications is provided.

7.1 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 [Dix,98, 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 [Miller,68], [Nielsen,94]. 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 [Bergman,00].

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 [Krüger,01]. 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.

7.1.1 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 summarised 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 [Norman,88] 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 [Shneiderman,83], 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 [Shneiderman,83], [Dix,98, 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 and disappearing interfaces. New interaction paradigms are required to realise 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 [Weiser,91], [Weiser,98]. Also the idea of a disappearing computer [Wejchert,00] 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.

7.1.2 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.

7.1.2.1 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 modelling this knowledge, knowledge representation, and to make this knowledge accessible for machines has influenced many approaches in research in robotics and artificial intelligence [Russell,95]. 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.

7.1.2.2 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 recognise 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.

7.1.2.3 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 [Lenat,98] 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 recognise 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.

7.2 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 characterises the new paradigm of implicit human computer interaction (iHCI). In this thesis 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 [Gellersen,99a], [Schmidt,01a], [Gellersen,02a].

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, recognised 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 centring the interaction in the physical environment rather then with

the computer system. A similar concept called “incidental interaction” is introduced in [Dix,02].

Realising implicit input reliably as general concept appears at the current stage of research close to impossible. A number of subtasks for realising 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.

7.2.1 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 recognising 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 recognise 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 of 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.

7.2.2 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 [Mankoff,01]. This puts the question:

- how to deal with ambiguities in iHCI?

7.2.3 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 25 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 recognised 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.

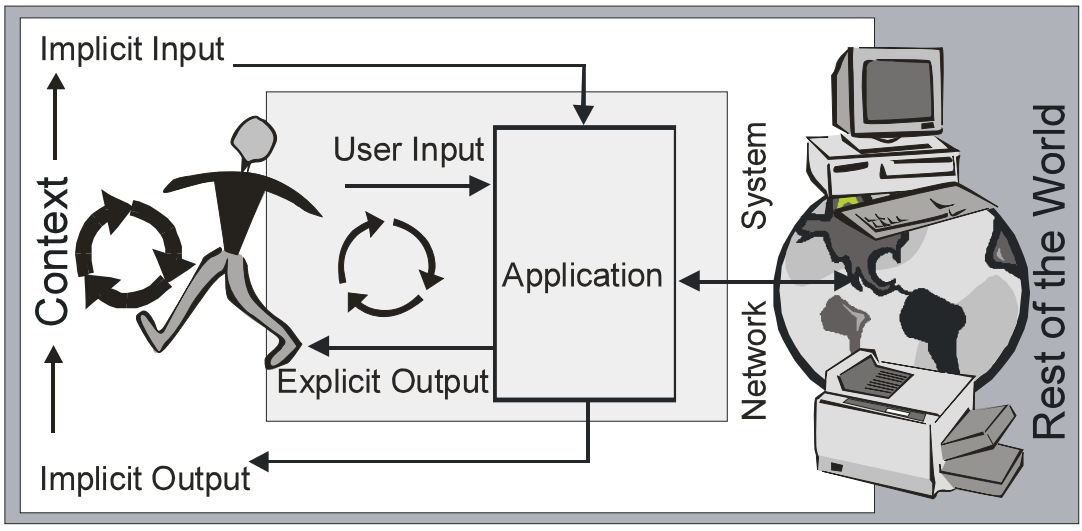


Figure 25: Implicit human computer interaction model.

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 centred 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.

7.3 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.

7.3.1 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 [Schilit,94], [Brown,97]. 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 recognised, 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 centred 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 [Cheverst,01]. 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.

7.3.2 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.

7.3.2.1 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 realised that context is a key concept for determining the appropriate configuration [Beigl,98]; a similar observation is also

reported in [Pham,00]. In particular to implement a system where the user is not surprised where the content will turn up is rather difficult.

7.3.2.2 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 [Schmidt,00a]. The orientation aware display described in [Schmidt,98] belongs also in this category.

7.3.3 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.

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

[Sawhney,98]. We showed similar findings in [Schmidt,99b]. 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.

7.3.4 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.

- 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 [Schmidt,00]. 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.

7.3.5 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 [Schilit,95]. 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 [Kirsh,95]. As demonstrated in [Beigl,00] this enables very powerful local communication paradigms.

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 characterised 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.

7.3.6 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

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.

Table 12: Using context meta data to retrieve documents.

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 12 examples are given of how context can be used to retrieve documents.

Meta information can become an important part of the data stored. Applications that automatically capture context are central to the idea of Ubiquitous Computing [Abowd,99] 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 [Thede,01].

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

7.4 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 [Cheverst,01a], but it also addresses the resulting implementation issues on system and application level.

7.4.1 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 [Intille,02]. 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.

7.4.2 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 [Salber,99] or to use broadcast as a general way of distributing context [SMART,02].

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. 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.

7.4.3 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.

7.5 Humans and Invisible Computing

The notion of invisibility and disappearing computing is common to the vision of Ubiquitous Computing as discussed in chapter 2. 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.

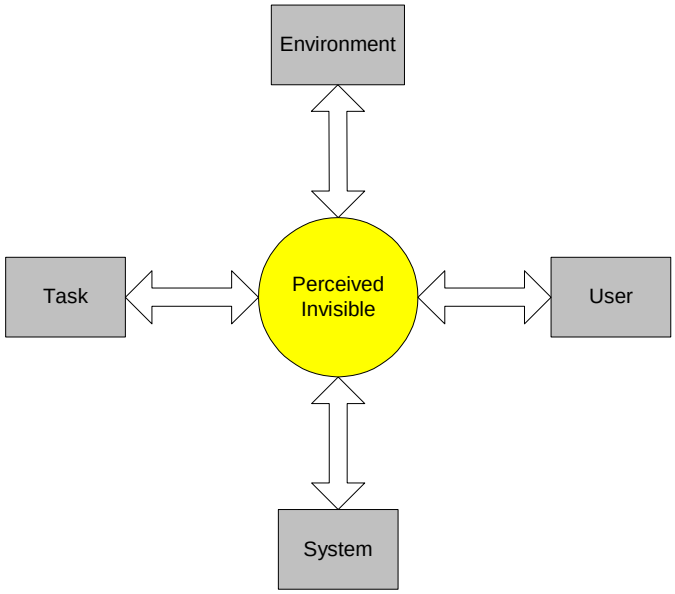


Figure 26: Factors that influence the perceived invisibility.

7.5.1 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 26.

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 [Norman,98,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. 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" [Norman,98] and the idea of "weave themselves into the fabrics of everyday life" [Weiser,91] 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.

7.5.2 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 [Gellersen,02] 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 visualised). 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 [Want,92], MediaCup [Beigl,01]) and they will be used as such – and not necessarily in a similar way as their non digital counterparts (a badge and a mug).

7.6 Discussion

In some models and implementations context is seen as just another form of user interface component that provides information to the system [Salber,99]. 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 [Mankoff,01] 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 memorising 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 memorise 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.

7.7 Summary and Conclusion

In Ubiquitous Computing 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.

Chapter 8

Evaluation

8.1 Evaluating Ubiquitous Computing Systems

The evaluation of Ubiquitous Computing systems is not yet fully understood. Researchers, with their own roots in different fields, seldom share a common understanding of how evaluation has to be conducted. In many cases even the very basic question what to evaluate can lead to controversial discussions. In research projects and publications it can be seen that often sub-parts of a Ubiquitous Computing system are evaluated with well known methods from well researched fields, rather than the whole system. In context-aware computing evaluation techniques for sub-parts are also borrowed from other fields. In particular the context acquisition systems are often evaluated using methods known from AI and neural networks; whereas the context aware user interfaces are evaluated with standard HCI methods.

It is recognised that evaluation of sub parts is essential to ensure that components fulfil the requirements. But it is also clear that evaluation of parts of a system does not necessarily provide an overall assessment of the system. Recently there is a growing interest in understanding specific evaluation problems that arise from Ubiquitous Computing systems [Scholtz,01], [Dey,02] [Scholtz,02].

8.1.1 Basic Evaluation Problems

When looking at Ubiquitous Computing systems and in particular context-aware systems there are still very basic problems. Some of those problems are just an issue the evaluator has to be aware of whereas others are more fundamental and require re-thinking of evaluation methods.

8.1.1.1 Evaluation in Context

Context-aware systems are designed to develop their full potential in context. Systems are designed to help users in a certain situation and provided information that is useful for a particular task. To practically evaluate such a system it is preferable to evaluate them in context. However this is often not easy as a situation may not happen very frequently. By forcing a situation to happen the usage experience is probably changed so that the result does not reflect the real usage in this situation.

Consider the following example experienced in the project TEA. In the interaction design it was identified that the mobile phone should show a different behaviour when there is an incoming call and the user has the phone already in her hand. A case that is rarely happening but it still may be of value. Doing an evaluation and giving the user a phone to use it over a period of two weeks it is unlikely that this situation will happen and therefore an assessment of the usefulness of the feature by the user would not be possible. There are two straightforward options to solve the problem, either prolonging the experiment to make sure that the situation will happen (at least with a certain probability) or to force the situation. The first case is usually not practicable when situations are unlikely as user studies would take years. And the second one may change the results as the user's expects the situation.

This problem is inherent to context-aware systems. It is necessary that this is taken into account when planning the evaluation of such systems. Depending on the system and other requirements a strategy has to be developed.

8.1.1.2 Multi Causality

By setting up a Ubiquitous Computing system in an environment, often many changes are made. In many cases this means that new devices are deployed, context acquisition systems are set up, new interaction metaphors are introduced, and so the

environment is changed. However from an application view only a new application is introduced, where all these parts play their role.

When now the application is evaluated it is hard to tell which part has played the major role. In conventional interfaces it is often possible to separate changes and do evaluations for each of the changes. E.g. to assess the influence of font size and colour of a label in a button it is possible to vary both independently and assess the implications on the user's behaviour. In Ubiquitous Computing this is often not possible. In some cases many components have to be introduced in the first place to enable certain functionality. In the evaluation it is then not possible or very hard to find the contribution of a particular design decision for the success or failure of the overall system. When evaluating Ubiquitous Computing systems it is important to minimise the number of variable elements to make it possible to figure out what effect is caused by which decision.

8.1.1.3 Evaluation Goal

In conventional systems it is often clear what the overall system is evaluated for, most often the goal is to proof that a system is more efficient. Variables associated with efficiency are the time to complete a given task or the number of errors that have been made while fulfilling the task (e.g. when writing a document). In Ubiquitous Computing when a system augments an environment enabling a user to do new things or to make boring tasks more interesting or more pleasurable, the metric is not straight forward anymore. In some instances this is similar to the evaluation of designs.

In cases where the user is empowered to do something novel which was not possible without the technology it is usually not possible or not useful to compare task efficiency. In cases where a task is made more exciting, pleasurable, or appealing the evaluation is always dependent on the subject doing the task. Therefore results will not be objective; however that does not mean they are not reproducible. If the group of subjects is well defined and large enough such evaluations can provide a very good insight.

It is important before evaluating a system to figure out what is the evaluation goal. Such goals can be to demonstrate the feasibility of a concept, show the ease of use, evaluate enhanced user experience, proof the efficiency or stability of an

implementation, and estimation of administration effort. All these issues require different evaluation procedures.

Beyond the evaluation goal it is also a central concern what to evaluate. Ubiquitous computing research includes more than prototypical systems. Theoretical concepts, system designs, user interface concepts, simulations, demonstrators, and deployed systems have to be also considered when discussing evaluation. It is obvious that there is not a single method, but it is important to identify what should be evaluated before doing it. In the following section some appropriate evaluation methods are discussed.

8.1.2 Methods Used

In this section a number of methods and approaches used to evaluate Ubiquitous Computing systems are presented. As there is not yet an established evaluation framework the following description aims at raising awareness and is open to further discussion.

8.1.2.1 Pre-implementation Evaluation

In many cases the design of a complete system can cost substantial effort. Moreover implementing and deploying systems is yet more costly. In cases where the feasibility and usefulness of systems is still in questions it is appropriate to evaluate before doing a full-blown realization.

At first sight it looks rather difficult to evaluate a system that does not exist. In many cases however a method known from other fields – called “Wizard of Oz” – can be helpful, [Dahlbäck,93], [Salber,93]. Instead of implementing a system one or more humans are used to mimic the system intelligence and interacting through interfaces which leave it open to the user whether there is a computer or human behind. Basically the computer’s behaviour is mimicked by a human to save implementation time [Maulsby,93].

If the system performance and the user experience is not as expected this is an indication that the implementation of the system may be rather hard and that it is useful to rethink the concept. It has to be mentioned that the human is used to mimic components which involve tasks at which humans are better than computers.

Examples are as language understanding, situation recognition, and prediction of behaviour. It is obvious not to let humans do tasks where computers are more useful (e.g. database lookups).

8.1.2.2 Sub-system Evaluation

The currently widely used practice of evaluating parts of the system based on well know methods from the field where these parts belong to is an important step. If the system includes a new networking protocol it is obvious that this should be evaluate with methods used in networking and compared to the metric used in this field. In most systems it is fairly straightforward to identify these parts and to use appropriate evaluation procedures. In project TEA we exercised this for hardware and for the recognition algorithms [Schmidt,99c].

Further examples of useful evaluation techniques are:

- simulation and test runs for network protocols
- usability test for new user interface concepts
- data collection and recognition performances analysis for context acquisition
- prototyping and demonstrators to proof feasibility of system design

Evaluation of sub-systems is essential but not enough. To evaluate complete systems further steps have to be taken.

8.1.2.3 Overall System Evaluation

When evaluating the overall system many different ways can be pursued, but it is essential to keep the basic problems describe above in mind. Depending on the system and the evaluation goal a strategy to include context has to be found. Furthermore it is central to minimise the number variables that have an influence on the evaluation to acquire useful results. The following approaches provide means to evaluate the overall system.

Single domain focus

The system is evaluated from the view of a single domain, such as human computer interaction or embedded systems. This is similar to the evaluation on sub-system level but includes further issues that appear in the overall system. In this approach only in the domain in question changes are made, everything else is kept constant. This evaluation helps to identify new issues in this specific domain and also offers a means to provide proof of advances made.

System Feasibility

The purpose of such an evaluation is to provide a proof of concept for a particular system. Usually by building a demonstrator it is shown that the implementation of functionality can be realised. Feasibility is however more than just showing it is possible; most often the statement is also that it is feasible given certain conditions such as overall cost, complexity, device size, and development time. Beyond showing feasibility, this type of evaluation also helps to gain experience while realising the systems. In many case the research path taken to show feasibility, including all the dead ends, is as valuable as the proof of concept itself. When proofing feasibility not everything must be implemented – parts that are obviously simple to implement, or that have been implemented before, are often taken for granted and not included. As this strategy speeds up the process and allows concentrating on novel issues it still bears the risk that the overall system may not be feasible after all.

Prototyping

To show major issues in a system it is necessary to build prototypes that are similar to the envisioned system. In many cases this includes that prototypical systems provide a similar usage experience as anticipated for the real system. The advantages are twofold. First, this makes it possible to experience the system as it is intended and can therefore give more insight on different usability issues. Secondly, by adhering to real conditions (e.g. power constraints, weight constraints, cost, and size) to some extent (including effects of future developments) the experience gained by implementing prototypes will be very useful to create the final system. Abstracting from these conditions while prototyping can make the life much easier but it often just delays tackling the difficult aspects. Prototypes can be used as tools to communicate with

potential users – and it offers communication in both ways. This issue of probing is addressed later.

Living Lab

Having prototypes – even with their limitations – in everyday use can give substantial insight into new technologies. The term living laboratory is used for different ways of deployment. When using this approach it is important to explain the characteristics of the experiment. The following questions raise some issues.

- Who is using it? Is it only the developer or also her colleagues?
- Is the prototypes shared or are there different prototypes available?
- How long is the prototype used?
- Who is administering and fixing the system when it breaks?
- How is the data from the experiment gathered? Is it complete? Did it interfere with the anticipated normal use?

When using a living lab evaluation it has to be kept in mind that this is always very subjective, however in many cases – especially with early prototypes – it is the only practical option to get experience.

In several projects at TecO and Lancaster the artefact and systems where used in a living laboratory style. In following some lessons learned are reported:

- **Stability.**
  As people work in a laboratory it is important that the prototypes work – at least with their basic, not enhanced, functionality. If things are awkward to use, the experiment is more like probing colleagues with a new system than using it in a living lab.

- **Novelty Effect.**
  New gadgets are always exciting, but how are things used in the long run? In a living lab environment it is important to keep things around for a long time. To

evaluate a system it is important to discriminate between the novelty effect of having a new gadget and the added value of a new artefact.

- **Generation Incompatibility.**
  Often prototypes evolve over generations. As knowledge is gained systems are improved or at least altered. These changes lead to incompatibilities between artefacts (e.g. the protocol on the MediaCup generations). To resolve this all systems have to be updated. A precaution is to include version numbers into protocol, even if it is not intended to have a second version.

- **External Dependencies.**
  If systems are used over longer term different problems occur. As systems are not isolated changes in other parts of the system (even beyond the reach of the developer) force changes in the system. A typical example is the usage of a web service (e.g. cinema booking system) where the web server implementation disappears (e.g. the company providing it goes out of business). To keep the system running a reimplementation of a subsystem is required.

- **Maintenance and Support.**
  Prototypes are most often not maintenance free, even if the envisioned system will have this property. In these cases it is important to have people taking care of maintenance and support – even if it is to support a "poster application" on the wall [Schmidt,01a]. Maintenance is a time critical issue. It is important to get systems quickly going after a breakdown to keep people using them.

- **Incomplete Systems.**
  When starting to build a system it is obvious that at the beginning this system is not complete. However it is often useful to already deploy working parts of the system in the living lab. This requires that parts of the system are built in a way that they can run without the rest of the system and optimally that they already provide some benefit, even in an incomplete state.

- **Compiling Results.**
  It is a central question how to acquire results from a living lab experiment.

Usage statistics, interviews and questionnaires are standard ways of getting this information. In many cases having questionnaires or interviews along the experiment (e.g. 5 questions every week) to establish the change in use is also very beneficial. However one has to be aware that by selecting and posing the questions the outcome of the evaluation can be greatly influenced.

Deployment and Studies

When systems are mature in technology or build on top of off-the-shelf technologies, deploying systems with genuine users is a very good option. By deploying systems in real usage issues that are central to the application will turn up. The great challenge here is to keep the system running over the time of the experiment. This is not just a problem of system stability this is also a question of discipline. Often in the very early stage of the deployment it becomes apparent that a slight change would improve the system. Then by doing the slight change new issues come up. This may be a good way to improve the system but in terms of evaluation it makes it much harder. Another challenge is to get the results from the users who participate in the experiment as the immediate value for them is often limited.

Deployment of systems has a further ethical dimension. Especially in the care domain if systems are deployed for a longer period and prove valuable to staff and patients it is difficult to take them away again. Often project resources only allow the administration of the system for the evaluation period. After this period the systems are removed, returning the environment for staff and patients back to the old situation. When planning such a study where systems are deployed in a real environment this must be taken into account. A minimal requirement is to make potential participants aware that the system is temporary and that they may have to go back to the “old” system after the trial even if it is successful. Another option is to plan to keep the systems running in case the trail is successful. These issues have emerged within Equator [EQUATOR,02] where there was a discussion to replace a paper wall calendar with an extended electronic version based on a Smart-Board.

As the discussion above shows many of these concepts are different form classical evaluations. One issue that appears over and over is that the purpose is to acquire

experience with the development, usage, and deployment of Ubiquitous Computing systems.

8.2 Evaluation of prototypes

As presented in chapter 4, prototypes have been a central tool while conducting the research leading to this thesis. Many prototypes have been built and a number of them have been evaluated in more depth. Two exemplarily evaluation efforts are presented here: probing prototypes and quantitative evaluation.

8.2.1 Probing Prototypes, Probing Concepts

Communicating novel ideas and concepts to the non-technical and non-research community is often very difficult as there is often little common understanding. However, getting feedback of potential users and ultimately buyers at an early stage is very beneficial for the development of technologies. If there is no communication with potential users there is a serious risk that research will explore issues that are of no interest to anyone. On the other side, potential users will often not consider their needs and requirements because the technology is very abstract and rather recite ideas from the science fiction genre.

The idea is to use prototypes and demonstrators as communication medium. The researcher can show what the technology is and what it can do for the potential user. And the potential user can imagine this technology in her life and assess the impact on everyday tasks. This is even possible if not all functionality is implemented.

In the project TEA this concept was used in a very early stage of the development. A workshop was held where an interested audience of potential users were invited. First, the audience received an introduction to the overall theme, in this case the idea of context-awareness, and then the prototype was demonstrated. People were allowed to discuss with researcher the prototype and its functionality. Often the discussion centred on every day scenarios of the visitors (e.g. an architect visiting a number of building sites a day), placing the technology virtually in their context. As potential users imagined the technology to be used in their context, questions about the functionality came up and where given to the researchers (e.g. “Could the phone know that it rains or that I am in the car?”). Based on these questions the users constructed

their own scenario which had value to them (e.g. “When I am visiting a building site and it is raining I don’t want to take phone calls. I rather want to call back when I am in the car again.”). These discussions were very fruitful for the researchers and fun for the users.

The workshop was concluded with a questionnaire asking about the technology and its implications. However, it showed that the questionnaire, as it was written by the researcher with little knowledge about the users needs, provided very much the information that was expected. In contrast the individual discussions in which technology was virtually put into people’s lives, provided more interesting results. The questionnaire and the results are documented in a deliverable for the project TEA ([TEA,98,D4.2, not public]).

People who are not involved with research but who are potential users of the technology often have a very different view on the technology developed. The results obtained from them may significantly differ from the findings of a living lab experiment. As in the living lab environment everyone knows what the purpose of the system and the technology is, thinking goes along these lines. When probing concepts and prototypes the potential users know much less and therefore will in many cases think more freely about the technology and its application.

In different projects, within the equator initiative, probing technology and concepts also proofed to be helpful and inspiring. Here again prototypes or just representations of prototypes [Boucher,01] provided a good way to communicate between different disciplines, namely designers, ethnographers, and technologists. By putting the concept into a prototypical representation (even if this is only a mock-up or visual representation) the imagination of the counterparts was stimulated and led to fruitful discussion and further development.

8.2.2 Qualitative Evaluation of Prototypes

When analysing the problem domain and also during the design phase often a number of requirements are identified and stated, this can also be called the specification. These requirements relate to the expected functionality of the system, but also may follow design decisions which were made for practical or aesthetical reasons.

In the qualitative evaluation the central question is: does the prototype meet the specification? This can be extended beyond the binary decision into a more detailed analysis. How well are individual requirements in the specification met, and what is the cost for meeting these specifications. When matching requirements versus the actual implemented prototype, the cost assessment is of particular interest. To further develop the prototype it is important to know which of the requirements introduced the major cost. This is then also view from the opposite prospective by asking how much easier, cheaper, more robust, or smaller the system would get when a certain requirements would be dropped.

Using the experience and knowledge gained by prototyping the system it is often useful to revisit the specification, possibly restating requirements. In some cases it becomes apparent that certain requirements can not be met and in other cases further requirements can be added to the specification as they have to be appeared useful in the prototype. However, this already shows a basic problem with qualitative evaluation. As the specification is not fixed people are tempted to produce specification after the implementation of the prototype matching the specification to the implementation which does not provided any insight or evaluation at all.

The following example shows exemplarily how a qualitative evaluation is carried out. In the project Equator we were interested in the tracking of people in particular where they are, how they move, and also detecting accidents such as falls. These objectives evolved after considering accidents statistics showing that most accidents of elderly people are falls, that people fall repeatedly, and that most falls happen in the home environments [Fuller,00], [NCHS,02]. The central functionality is to provide a sensor that can be used to alarm others when a fall has been detected. And further more to monitor a history of fall to make medical interventions possible. The main requirements are stated as follows:

- detection of position
- detection of falls
- calculation of the overall movement

- suitable for all rooms in home environments
- unobtrusive
- privacy preserving
- robust implementation

The load floor, see section 4.4.2, is prototyped using load sensing technology. Each of the requirements is revisited and the feasibility is assessed based on the actual implementation. Considering a single person in the environment it appeared that the detection of position and falls is fairly easy to do. However for multiple people this is becoming much harder and potentially impossible. In coherence with the original scenario the requirements are restated to a single user scenario. The reports of falls with elderly also revealed that many of the accidents happen in the bedroom or in the bathroom therefore privacy becomes an important issue. As privacy is not an absolute value it has to be compared to other options available (e.g. Granny Cam, [Greene,02]), and here it shows that the load based solution preserves privacy much better.

This shows that the load based tracking solution passes a qualitative evaluation of the requirements stated in the first place, but with the restriction to a single user in the space. In the scenario envisioned this does not jeopardise the usefulness, as the benefit of the system is to people when they are alone.

Qualitative evaluations are a standard method in practical computer science where systems are designed and build. It is apparent that this way of evaluation is subjective; however in many cases it is the only possible solution to evaluate the prototype of an overall system.

In this thesis the main arguments in chapter 3, 4, and 5 are concentrated in hypotheses. To show how evaluation of context aware systems can be done in the reminder of this chapter each of the hypotheses is revisited.

8.3 Revisiting the Hypotheses

Four hypotheses are stated to highlight main points that are investigated in the course of this research. The first looks at the very basic question of context acquisition using sensors, the two following are concerned with modelling of context starting out from artefact, and the final one is concerned with the prototyping of context aware systems. In the following each of the hypotheses is stated again and analysed. Than arguments that provided evidence that support the hypothesis are presented.

8.3.1 On Context Acquisition

Recalling the fact that in the real world all situations are different it is interesting that from everyday experience humans still think of the "same thing" happening again and again. When assessing a situation not all information is taken into account, only the information that is characteristic or discriminating. In the terminology used here an abstraction from situation to context is implicitly carried out.

Waiting in a queue to pay is an example of a situation familiar to many people. The context "waiting in a queue to pay" can be found in very different settings, just consider different types of shops, different types of goods purchased, in different countries and cultures, different methods of payment, and different times of a day. Still humans have little problem to find out whether or not a specific situation fits that description. This indicates that only the subset of characteristic features is used to assess whether or not a situation fits a context. This leads to the first hypothesis, which is already stated in section 3.3.

<u>Hypothesis 1:</u> For all situations that belong to the same context the sensory input of the characterising features is similar.

In this subsection it is illustrated exemplarily how such an evaluation can be done. For a comprehensive evaluation a greater number of situations and more data samples would be used.

The sensory input originates from a sensing system, or more technically from sensors, which can be seen as changing variables over time.

In Figure 27 the raw sensor values for selected sensors for four different situations are shown (each context is 100 time steps long). The situations are described on page 51.The readings are taken from a general sensor board that is attached to a cup. The top graph shows all sensor values, the middle only light values, and the bottom one

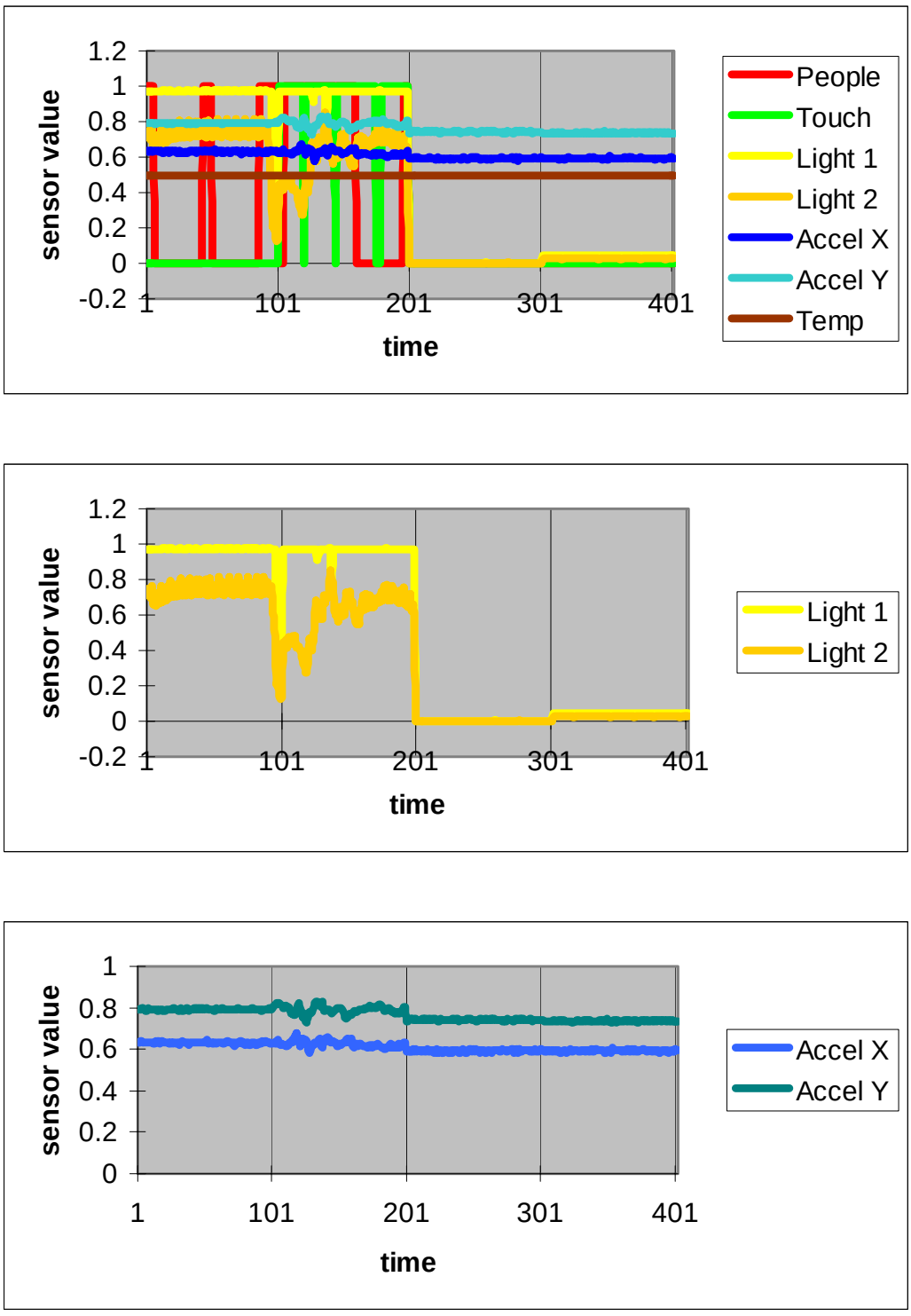


Figure 27: Time series plot of raw data.

acceleration data. Just from observing the data plot it becomes apparent that the situations generate different sensory inputs. However, it is also clear that not all sensors are equally helpful to discriminate the contexts and that a processing of the raw data into cues and features is useful.

To show that when selecting the characteristic features it becomes possible to discriminate situations into context the following procedure is performed. This approach has been used in various projects described earlier. In this section it is exemplarily carried out to support hypothesis 1.

After specifying all contexts that are of interest, sensory input in various situations belonging to these contexts is collected using physical sensing systems. Based on the domain knowledge and from analysing the data characteristic features are selected. For each of the contexts a set of typical stimuli is calculated and stored. For any new situation the same stimuli is calculated and matched against the representative stimuli. The fact that the difference between the stimuli based on characteristic features is much smaller than over a larger number of calculated stimuli provided evidence that supports the hypothesis.

In this evaluation this is shown using the following example. The following four contexts were selected and data in various situations belongs to these contexts have been recorded.

- Context 1: cup on table while person is working on the desk, the office light is on, the cup is not touched
- Context 2: user holds cup in her hand thereby touches the handle, as the cup is used it is moved
- Context 3: the cup is in the cupboard and the cupboard door is closed
- Context 4: the cup is in the unoccupied office, only emergency light is on, no one is around

To discriminate the situation the following cues have been identified as characteristic:

- Cup touched or not (capacitive touch) over a 3 second interval
- Someone moving in the space (passive IR) over a 3 second interval
- Light level (averaged both sensors over the last 3 seconds)
- Accumulated absolute change in acceleration over a 3 second interval

To compare to a richer stimuli the following additional cues have been calculated and used to assess similarity.

- Accelerometer value in x and y direction (averaged over the last 3 seconds)
- Current temperature (averaged over the last 3 seconds)
- Change in light over a 3 second interval
- Light level of each light sensor (averaged over the last 3 seconds)

For each of the contexts, data is recorded and a representative stimulus on all cues is calculated, normalised, and stored as a sample vector. New situations belonging to one of the four contexts are recorded and the normalised cue vector is calculated. The vector is then compared to the four sample vectors and thereby the closest context can be found. In Figure 28 the average distance for calculated vectors from 48 situations recorded in a certain context (12 situations in each context) and the sample vector for a context is depicted. The top diagram shows the distance based on the small cue set and the bottom one based on the extended one. The distance to the context where the situation belongs to is always the smallest. It is interesting to see that the extended cue does not always perform better, e.g. the difference between the sample vector for context 4 and the recorded vectors for situations in context 3 is better with the small cue set. By increasing the number of stimuli – and not selecting characteristic features – the process becomes more depended on the learning and generalization ability of the matching algorithm.

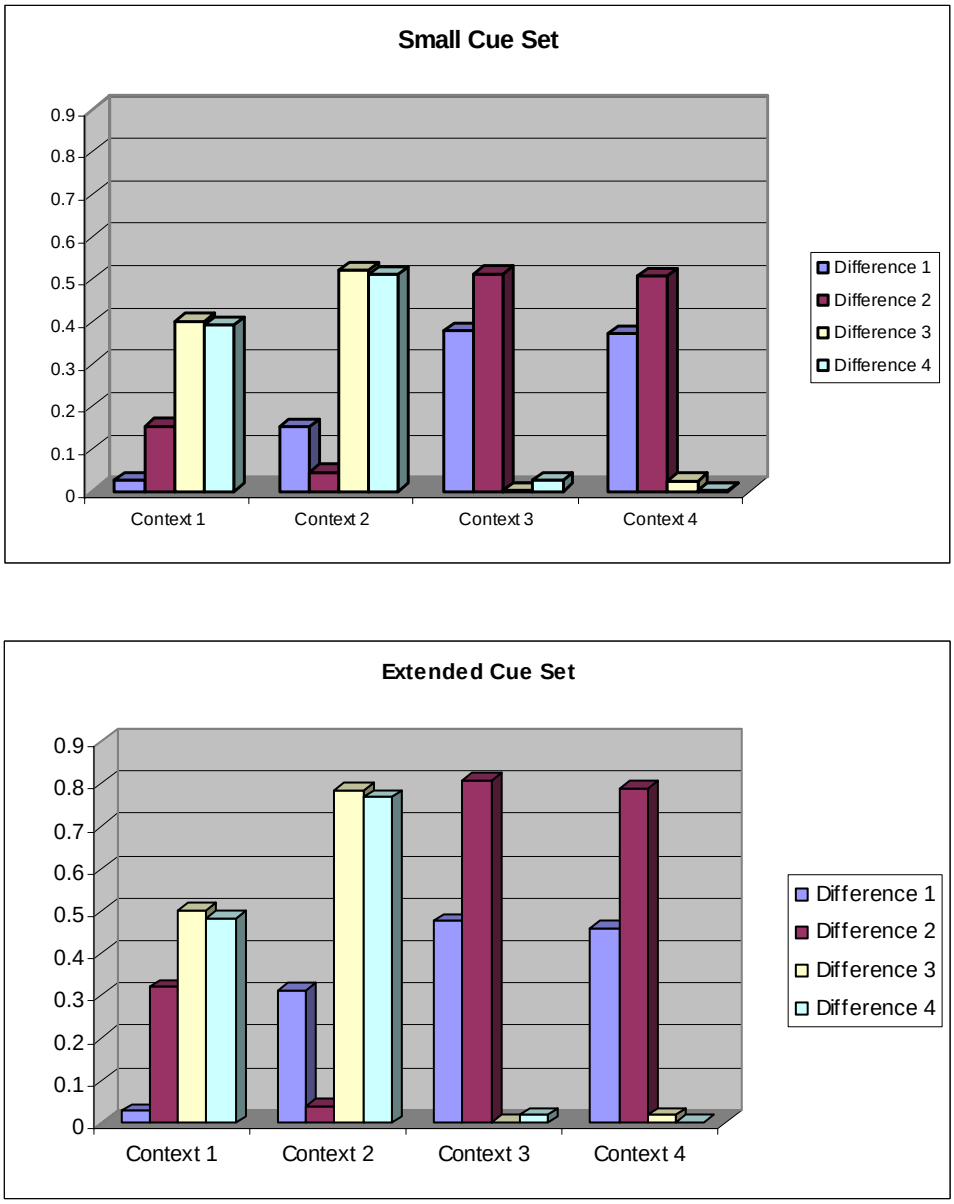


Figure 28: comparison of characteristic features calculated for each context.

Similar to this experiment a number of other experiments supporting the hypothesis have been carried out, [Schmidt,99c], [Schmidt,02], [Beigl,02]. These experiments do not proof the hypothesis but provide strong evidence that it is correct.

8.3.2 Context Modelling

When modelling complex systems there is always the choice where to start. The experience gained in building prototypes suggest that for context acquisition systems it is useful to start at the artefact, doing a bottom-up model rather than a top-down model. This is discussed in section 4.2 and leads to the following hypothesis.

Hypothesis 2: The domain knowledge about a specific single entity is more universal and easier to establish than the domain knowledge of a complex system, and hence it is simpler to identify and implement contexts on entity level than on system level.

This hypothesis comprises multiple claims:

- the domain knowledge about an entity is more universal than of a complex system
- this domain knowledge is also easier to establish
- and as it is simpler it also becomes easier to implement

When looking at artefacts in everyday life there can be found a strong support for this argument. Consider a wine glass – the domain knowledge of this artefact is greatly independent of the surrounding it is used in. The basic actions people do with a wine glass (e.g. pouring a drink in, drinking from the class, cleaning it, and storing it in some place) is independent of whether the class is used in a home setting, in a restaurant, at a banquet, or on a ship. The basic domain knowledge will hold in most cases without any knowledge about the surrounding situation.

This approach was successfully used to identify the basic context relevant for different artefacts in a number of projects, such as TEA, load sensing table, and MediaCup. When identifying the basic context for objects very little disagreement arose whereas when identifying contexts on system level it is hard to agree on a common set of contexts. Within workshops this approach was used to identify artefacts and potential relevant contexts. It showed that after identifying artefacts relevant in a given environment there was in most cases a general agreement on the basic dynamic properties. These dynamic properties related then to contexts.

To support the hypothesis a short account of a particular workshop is be given here. The workshop was held at TecO, University of Karlsruhe, on the 27th and 28th of June 2002 (The results of this workshop are not published yet). From different

European projects15 people were invited to participate. All participants had a computer science background, but with different fields of expertise including HCI, computer networking, AI, neural networks, and embedded systems. The prime goal was to create a selection of aware artefacts by specification. The specification included:

- the contexts that are specific to a particular artefact,
- the sensing infrastructure that would be need,
- the processing system and algorithms,
- and the communicating requirements.

Participants worked in groups of 3 to 5 people. Each group selected artefacts which they found interesting and created the specification for those. Examples of artefacts assessed are: table, chair, cushion, cup, book, pen, phone, sink, door, door handle, bottle opener, ski, and bike.

The results and the experience reported by the participants strongly supported the hypothesis. In most case a strong agreement of the main contexts relevant for an artefact was reported. Similarly after having found the relevant contexts the specification of the implementation was straight forward, given the selected contexts can be implemented by regarding basic side conditions on sensors and processing (mainly defined by the Smart-Its platform).

Given that contexts for many artefacts are available the next hypothesis becomes very interesting and central for building useful systems.

> **Hypothesis 3: Contexts for an entity or a group of entities can be established by fusing the contexts of entities that make up the entity or the group. Thereby artefact centric context enables versatile uses and becomes the foundation for a platform for applications.**

The hypothesis claims that new contexts can be created when contextual knowledge is already available and that this leads to more flexible use. In different projects where contexts of artefacts have been made available new applications evolved. These applications often use contexts that are related to artefacts or environments that do not supply context themselves. These contexts are established by fusing contexts of artefacts that are available and spatially or logically related to the artefact or environment of interest.

To support the hypothesis a closer account of the contexts provided in the project MediaCup is presented. Each cup provided the following basic contexts:

- cup on the table
- drinking from the cup
- playing with the cup
- temperature context (hot, warm, cold)

Contexts are communicated regularly and also include an identity and coarse location information. The contexts are communicated via IR to the backend and the gateway propagates them via UDP-broadcast into the local network.

This information is then available and can be used to fuse higher level contexts that may be of interest to other applications. Especially the information of co-location and co-occurrence of contexts provided effective means to calculate further contexts. Examples of new contexts that evolved in the MediaCup are: “a room is used”, “people having a meeting”, and “chat in the hallway”, see Figure 29 for an application of the second context.

Looking at these examples it is also apparent that the reverse reasoning is not valid. It is not possible to rely on these sensors to detect a meeting. In just a simple case where people don’t take a cup with them the system would not work. Nevertheless having information of sub-parts available makes it feasible and useful to calculate new contexts. In an environment with a large number of aware artefacts and hence with a lot of context information, fusion of new contexts becomes a very powerful tool.

Similar to the MediaCup experiment it could be observed that providing basic context can lead to the creation of new contexts based on these. It is also interesting that the new contexts often are not envisioned by the one developing the original contexts for the artefact. However as these contexts are related to artefacts and therefore easy to understand it is easy for developers to built on them and create new contexts, often more abstract.

8.3.3 Rapid Prototyping of Context Aware Systems

To make it feasible to create many artefacts that provide context it is inevitable to ease the development and implementation of context acquisition systems. In chapter 5 the argument for a rapid prototyping platform is made, stating the following hypothesis.

> **Hypothesis 4: A rapid prototyping platform to make artefacts context-aware can be provided. Such a platform will simplify and speed up the prototyping process of such systems.**

This hypothesis contains multiple claims, first that it is possible to build such a platform, second that this will simplify the development of context aware systems,

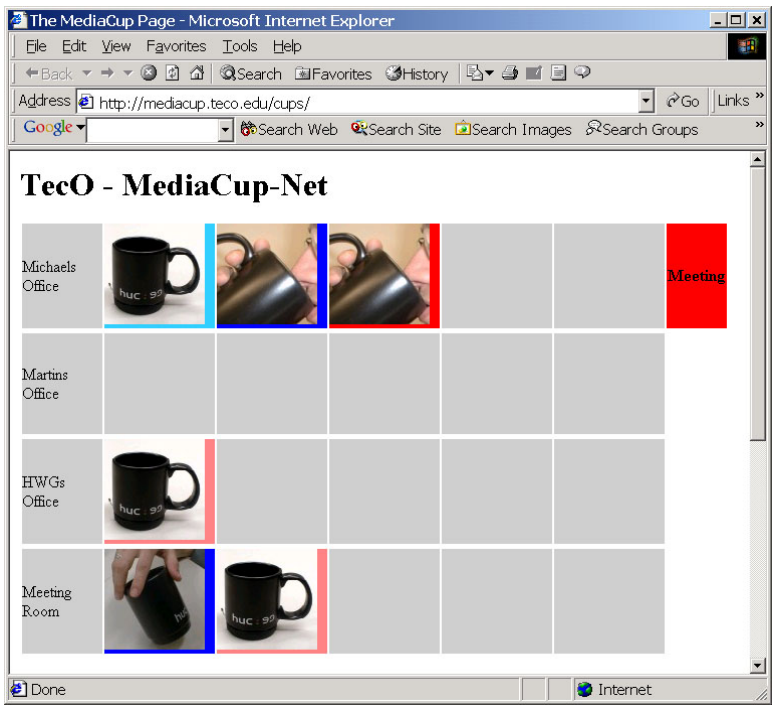


Figure 29: calculated high level contexts based on simple MediaCup contexts.

and finally that this results in a quicker development process. To support the hypothesis it was obvious that such a platform had to be designed, built and evaluated.

To validate the first claim building the platform and ensuring its functionality can be done by a qualitative evaluation matching the requirements with the actual implementation. However, in this research the second and third claim are of more importance. To support these claims it is clear that the platform has to be used to develop context-aware applications and to analyse the development and implementation process and comparing it to others ways of implementing such systems. By doing so the first claim is implicitly proofed, as without a platform this would not be possible.

To assess rapid prototyping of context aware systems based on the platform developed a developer's workshop was organised. The aim of the workshop was to get people together who are familiar with the idea of ubiquitous and context aware computing but who have a computer science background and have no or only a minimal knowledge about the design of electronic systems. The task for the participants was to develop a new context aware device, consisting of:

- context acquisition hardware,
- context acquisition software,
- communication,
- and backend application.

The two and a half day workshop was held in Lancaster. 15 researchers and students with a computer science background working on projects in the European disappearing computer initiative [DC,02] and in Equator [EQUATOR,02] were invited and participated. They came from 4 European countries, and 7 different institutions. Of the participants 14 were male 1 was female. 3 of the participants were completing their master, 9 working towards their PhD, and three were senior researcher. All but one of the participants had never designed and built a MCU system. About half of the people had programmed a microcontroller before. Several people had never soldered before.

The workshop was organised so that people worked in groups of two or three people together. In the first day the hardware for the core board and for a sensor Add-On board was build. To ease the construction of the boards a visual step by step manual was provided, see tutorial [Schmidt,02b] and the video documenting the atelier [Schmidt,02c]. To test the basic functionality (sensing and communication) software templates provided were compiled and programmed onto the systems. On average every participant built 1.5 core boards (22 in total) and 1.1 sensor Add-On boards (16 in total).

For the reminder of the workshop it was left to the individual groups to invent, design, and implement a context-aware system. Each group had at least three core boards and two sensor Add-On boards. This allowed two mobile units and one base station. The following devices were realised within the available one and a half days:

- Smart Ball
  A ball which includes a sensing board that recognises that it is thrown and caught. The ball communicates these contexts to the base station which is connected to a PC. When a context appears a sound is played. In this case the implementation played an explosion sound when a user caught the ball.

- Wireless Gesture Remote Control
  In this project a sensor board is used as a wireless gesture remote control. By turning the device the volume can be adjusted and with other gestures tracks are skipped. The mobile unit is wirelessly connected to a core board working as base station. The integration is realised using an adopted WinAmp plugin.

- Singing Smart-It
  The sensor board was extended by a piezo module to produce sound. Depending on how the object is handled it plays a certain tune.

- Wireless RFId Sensor
  One group decided to use one of the experimentation Add-On boards and put a RFId module on it. The RFId module is connected to the core board using the digital I/O pins but running a serial line protocol. The software developed on the core board communicates with the RFId module and sends the Ids that are

read back to the bases station by RF. Another core board is used to receive the data.

- Wireless Gesture Joystick
  In contrast to all other groups where the base stations were connected to the serial line this group decided to built an Add-On board that allows to connected the base station to the game port of the PC. Using a digital potentiometer the two axis of the joystick can be simulated. The mobile unit, a core board with a sensor Add-On, was used to identify gestures; these were then communicated via RF to the base station and translated into joystick movements.

One further project, a system that can detect the walking behaviour of a group, by having a sensing unit attached to each one in the group, was not finished in the time of the workshop.

These results and the response to the workshop indicate that the rapid prototyping platform is a very efficient tool for building context-aware Ubiquitous Computing applications. In particular participants were amazed how quickly they had working wireless sensing systems. The approach that each of the participants had to build the hardware from scratch – costing about a day – was questioned at first. However, in the course of the workshop people found it very positive and commented that this increased the understanding for the process and the system to a great extent. This also explained that people felt quite confident with the platform right from the start. Further comments also suggested that this may have speeded up the further development process significantly.

Furthermore, it was interesting to observe how people build the overall architecture of their systems. Specifically two dominant ways are pursued. A section of the attendees changed the code on the MCU only a little, mainly adding an identifier to their RF packets, and sending all sensor data at the maximum data rate to the backend. In this approach all processing was done in the backend. The other section of the attendees followed the opposite approach, doing as much processing on the mobile node as possible and only communicating contexts. This second approach is preferable in

terms of bandwidth use and power consumption. However the first approach saved a lot of time, as development is much faster for the backend system.

8.4 Discussion

The evaluation of Ubiquitous Computing systems is a central issue [Dey,02]. Different methodologies can be followed all of them have both advantages and disadvantages. In most case there is no single optimal way. Often doing different evaluations, based on different methods and techniques is the most practical way. With all steps taken it needs to be borne in mind that a prime goal of evaluation is to ensure the quality of research, particularly that the knowledge gained is valid and reproducible.

For Ubiquitous Computing systems it is central to discover what the objective of a system is and than look for evidence that supports the claim that the objective is met. The first step to find the objective is to realise what gets better, simpler, quicker, or more pleasant, with the system being proposed. In contrast to conventional systems this is not straight forward for most Ubiquitous Computing installations. It is only possible to do a useful evaluation when the purpose of the original system is understood. Where systems are at the design periphery, such as ambient displays, describing the functionality becomes very difficult, because it is beyond communication of information and aesthetics issues become central.

In the evaluation of prototypical systems there is also the question whether or not it should be abstracted from shortcomings of current technology. In many installations, where the prime goal is to explore the implication of a new user experience and a new relationship between the human and the computer, the actual technology used to prototype is not of central interest. Often it is assumed that at the point when such installations will be widely available the technology will have changed anyway. Under this assumption often the administration effort with current technology, the complexity, and power consumption is not assessed. However, in many cases including the shortcomings of current technology into the evaluation could give insight for the development of next generation technologies.

One central problem that goes beyond evaluation is how to communicate the knowledge gained by exploring and prototyping systems. Furthermore the question is

also how to extract the knowledge that is valid beyond the single case and is useful in further development. Many results which are published are taken from a very specific installation and therefore the results may be valid but in most cases they are not reproducible. Proofing the validity of a more general claim, abstracted from a single case or from a few applications, is in many cases hard or even impossible.

In this thesis the approach of probing people with technology is assessed. From the experience so far this helps that people can put new technologies into their context and consider the implications. As probes are specific and non abstract artefacts it avoids that people respond with science fiction fantasies which they have read or seen. Also having a physical artefact that has a certain functionality which people can explore and play with scenarios become more realistic. Compared to other methods this is a relatively quick way of getting feedback in early stages of the development. Running workshops where the exploration is extended over a day or even a few days improves the outcome; however this is a fairly expensive way to evaluate.

8.5 Summary and Conclusions

In the research carried out within this thesis different ways to evaluate Ubiquitous Computing systems and in particular context aware systems were explored. In general it can be seen that evaluation is in most cases not simple and borrowing methods from other fields is not satisfactory. In publications it is often seen that sub parts of a system are evaluated with well known methods but that the complete system is not evaluated due to the lack of a standard method. Therefore it is important that evaluation methods are developed for Ubiquitous Computing.

To show the potential range of how evaluation can be conducted, methods for the evaluation of systems in the design stage, of sub parts of systems, and in more detail of complete systems are presented. These methods have been explored in various projects carried out within this research work.

Motivated from the research carried out, a more in depth discussion on the evaluation of prototypes is presented. It is argued that prototypes are a very useful and powerful tool in Ubiquitous Computing research. In particular the idea of probing potential users with prototypes is explained. A method successfully used in different projects.

The experience gained, suggests that probing “ordinary people” with prototypes and using prototypes in a laboratory environment is orthogonal. From probing people real life issues, concerns, and possible new applications emerge. In lab usage results are more of technical nature and also common usability problems can be detected. In general probing can be done with early prototypes whereas in a living lab a certain stability of the system is already required.

Given this background information on evaluation the hypotheses stated in chapters 3, 4, and 5 are revisited. For each of the statements evidence is provided that supports the claims made. In two cases results from workshops have been used to evaluate methods and tools.

In discussions with researchers from fields where evaluation is formal and well understood the evaluation methods in Ubiquitous Computing are sometimes criticised. In contrast when comparing to evaluation in social science, design, and arts the way research is evaluated in Ubiquitous Computing is very acceptable.

Chapter 9

Conclusions

In this thesis research in Ubiquitous Computing and in particular in the area of context aware systems is presented. The thesis touches on many subjects, such as context acquisition, sensing and perception, modelling and prototyping context aware Ubiquitous Computing systems, development support, distribution, human interaction with computers, and evaluation. In retrospect, and having insight gained in the time since, it may have been wise to concentrate on just one of the issues, however when the research that led to this thesis started, in the beginning of 1998, the picture was less clear.

The rapid change in understanding can be illustrated by looking at the first hypothesis stated in chapter 3. Currently it is widely accepted to use different sensors to acquire context information. At the beginning of the TEA project this was still regarded a disputable research question, but considered worthwhile for funding. Similarly the understanding that “context is more than location”, as we pointed out in [Schmidt,98], and that different physical sensors can offer a significant contribution to the context information perceived by the system, has only evolved in the research community over the recent years.

Since then, the interest in Ubiquitous Computing and context awareness has changed dramatically. Looking back at the research carried out, the papers published, and on the contributions made, it is interesting to observe that certain issues, which we

regarded novel and an important contribution a few years ago, are now considered common sense. In the following section the main contributions and results are summarised.

9.1 Contribution and Results

The main contributions of this thesis are in three areas. Firstly contributions towards understanding the research in Ubiquitous Computing are made. Secondly architectures platforms, methods and tools are developed; and finally this is put into the context of humans using it.

9.1.1 Understanding research in Ubiquitous Computing

In this thesis steps are made towards understanding the nature of research in Ubiquitous Computing. Compared to other well established subject areas (e.g. neural networks, the area of my master thesis research) the approach is still in its early stages. In particular the following issues have been found

- In chapter 2 roots of research in Ubicomp are assessed and their influence on the research methodology used. In particular it can be observed that most often there is not a specific Ubiquitous Computing methodology. In many cases rather it is an adapted methodology that is strongly related to the background of the researcher (e.g. HCI, mobile systems, and AI).

- In chapter 4 the bottom-up approach to the design of context-aware systems is introduced. This research methodology relies strongly on prototyping and documentation of the experience gained in this phase. This methodology can be extended to Ubiquitous Computing in general. As prototyping is costly, methods and tools are required. This became a central issue, as introduced in chapter 5. These tools and methods can also provide a foundation for the reproducibility of research carried out.

- Evaluation of Ubiquitous Computing systems, as a major consideration for assuring quality of the research carried out, is assed in chapter 8. The contribution is firstly to compile a record of methods and techniques used and secondly to introduce the approach of probing people with demonstrators and

prototypes. Probing is orthogonal to most of the other techniques and aims to provide feedback to the developer in an early stage of the research. Also a report and example of the application of evaluation methods is provided.

- The quest for the "killer application" in Ubiquitous Computing leads us into the wrong direction. In the first review for the project TEA one important question was: what is the killer application of context-aware systems. Within the work of this thesis it became clearer that this is not a central question. Ubiquitous computing introduces incremental changes to many areas and hence enables new ways of interaction between humans and their environment and provides new dimensions for applications. The quest for the killer application distracts the attention from basic research that is required to get the infrastructure in place. It many areas it is a good idea to use applications which are obviously not killer applications but are useful to provide insight into certain problem domains, examples can be seen in [LaMarca,02], [Antifakos,02].

The issues presented in this thesis do not result in a coherent new research methodology. They rather provide a set of "tools" that can help to facilitate reproducible research. As the research field is very diverse and also the focus of researchers is often different it is inevitable to accept a broad set of evaluation methods, as long as they ensure the quality of the research.

9.1.2 Architectures, Platforms, Methods and Tools

A major part of the work was dedicated to the development of tools that support prototyping of Ubiquitous Computing systems and in particular the acquisition of context. Also methods, architectures and frameworks have been developed. However before stating methods and developing tools it is essential to understand the task that needs to be done and systems that should be built.

- In chapter 3 it is assessed how perception can be realised and what technical sensing options are available. This analysis is tailored to the domain of Ubiquitous Computing and in particular focusing on systems with highly restricted resources. This goes beyond surveying sensors and considers further questions such as learning and suitable abstraction. In chapter 4 issues on how

sensing is integrated with artefacts deepens these issues. The knowledge about sensing systems is then used to implement sensing building blocks, sensor drivers, and perception libraries, as shown in chapter 5.

- To build software abstraction for context acquisition systems, as presented in chapter 5 and 6 is inevitable to understand possible architectures of such systems. In chapter 3 a perception architecture is introduced providing different abstraction layers and interfaces between sensors, cues, contexts, and applications. Distribution is identified as a central aspect when contexts are related to artefacts in the real world. The model and platform introduced in chapter 6 takes spatial and temporal distribution into account. The contribution is to show how principles observed in human understanding can be used to create a model for Ubiquitous Computing and also how to make these principles a central part of the communication platform.

- In chapter 5 a method is introduced that offers support when building context acquisition systems. The method reassembles the knowledge acquired when building various prototypes. Tools have been built to ease certain steps. The method itself becomes the foundation for further developments; as it offers a structured model how to create such systems.

- Building context acquisition systems involves in most cases hardware, software, and communication. Up to now a standard sensing platform has not been available. In chapter 5 contributions are reported to ease this issue. Firstly building blocks and libraries for the most commonly used sensors are provided together with a modules (software and hardware) that provide processing and communication. Secondly a prototyping platform with readymade sensor boards that can be attached to processing and communication boards is presented. From the patterns of aware artefacts three most common architectures evolved. Given the hardware building blocks and the prototyping platform for each of the proposed architectures a software framework is provided.

- Ubiquitous computing systems are often complex and the development may be incremental. For many considerations (e.g. architecture in chapter 3, bottom-

up approach in chapter 4, Smart-Its boards design in chapter 5, and distribution platform in chapter 6) the support for the development process was a central concern. Especially the two questions how to simulate parts of the system and how to debug such systems had a great influence on decisions made. The models, architectures, and tools presented in this thesis are designed to support debugging and simulation on different levels.

It is shown that through the availability of tools, and the understanding communicated by models, architectures and methods, the process of prototyping context aware Ubiquitous Computing systems becomes simpler and quicker.

9.1.3 Interaction with the Ubiquitous Computer

The visions of calm computing, invisible computing, and the disappearing computer have in common the inclusion of a strong focus on how users experience new technologies, and environments augmented by computing technology. The "human in the loop" was always a central issue when thinking of models, architectures, and systems.

- Within the research for this thesis it became apparent that explicit user interfaces, independent of their modality, do not satisfy the properties on the human computer interaction stated in the visions presented in chapter 2. With this thesis the concept of implicit human computer interaction is contributed, as introduced in chapter 7. Implicit interaction aims at the interpretation of human behaviour as input to a computer system. Combining implicit and explicit interaction new, and more subtle, forms of human computer interaction become possible and can be realised.

- The usage of context, and more general of implicit input, in combination with explicit input is investigated. Different options, context push and pull, are discussed. Advantages and disadvantages depending on the requirements of systems are presented.

- The concept of invisibility of interfaces is critically assessed. In this thesis the term "perceived invisibility" is introduced to highlight that invisibility is not an absolute concept. The environment, the user, the task, and the systems are

identified as main factors that influence the degree of invisibility. Before designing and building user interfaces that should be perceived as invisible the influencing factors must be understood for each case.

The points mentioned above are central to the design of user interfaces for Ubiquitous Computing systems. However interaction with non-standard interfaces creates further challenges, which are addressed in the thesis, even so if there is no answer to some of the questions, providing awareness for the problems is a contribution in itself.

9.2 Future work

Usually with each prototype finished, each system evaluated, and each paper published a number of new issues that pose interesting challenges appear. Extending the vocabulary of the pattern language towards a semantic context model and creating an integrated tool that eases development of Ubiquitous Computing systems are two issues of central importance.

9.2.1 Towards a Semantic Context Model

In this thesis 9 patters have been presented based on the prototypes build. These 9 patterns are not a complete language yet. To make the language more expressive further words should be added. It is however essential that patterns, as they become a foundation, are validated, preferable by prototyping instances of them.

Having now a tool in hand, such as the Smart-Its rapid prototyping platform, creating new context aware artefacts becomes easier and quicker, and as we see already in our lab more new prototypes flourish. To communicate the knowledge gained in the prototyping phase patterns describing the experience should be added.

For a number of patterns it is also of interest to provide more detail. This can be either done be rewriting the patterns. In most cases specifying patterns that are just looking at a sub part of one of the patterns is preferable. To compare it to the original pattern language [Alexander,77] the current patterns are the types of “houses” and there is still a long way to go to describe “room” and “the parts that construct room”.

Extending the pattern language, describing the experience gained, and interlinking the vocabulary can eventually lead to a semantic context model based on a sound foundation.

The pattern language as introduces in chapter 4 is tailored to context and context aware artefacts and systems. It would be beneficial to extent it to Ubiquitous Computing systems in more general.

9.2.2 Creating a Physical Interface Toolkit

In this thesis building blocks and libraries are presented. Relating these to the architectures and to the method introduced it becomes feasible to provide tools that ease certain steps in the development process. Ultimately such tools can become a physical user interface builder for Ubiquitous Computing systems, which supports implicit as well as explicit user interaction.

- A tool that helps the developer to identify the contexts that matter for the application in mind can be realised based on an expert system. The output of this step is then a list of context. Interactively the developer can then select a set of variables and features to mach the requirements.

- The developer can specify architectural constraints, e.g. who is the context users. Based on the list of sensors, cues, and features, and also taking into the account further constraints, an architecture can be suggested. With this information interactively established the tool can provide libraries, software frameworks, and documentation tailored for this particular system.

To build an integrated development tool the key issue is to combine knowledge and building blocks from all the areas that have to be covered. The goal is to build a tool to create a complete hardware and software design of the device that can be given into production. The overall result of the system would comprise:

- A block diagram of the architecture

- A comprehensive hardware description of each unit in the architecture, including schematic, a PCB layout that fits the physical size, and a part list

- Software source code for each of the units.
- An API as specified by the designer that can make use of the contexts.

The input to the system is to specified by the developer and comprises following:

- The physical shape and size, in 3D
- The power source to use,
- The possible placement of the sensors, which requires knowledge about the physical design (where can be hole for sensors to peak out, where does motion happen).

From the experience gathered in the course of the research such a system is possible and could be incrementally extended to a physical user interface builder.

9.2.3 Further issues

The investigation of context and especially context in relation to communication and distribution is an area with many open questions. Initial investigations suggest that using context to address communication partners can be a powerful concept. The idea of "ContextCast" as a non-id based addressing extents the idea from location based addressing, as suggest in GEOcast [Navas,97] to context in general. ContextCast is based on the idea that proximity in a multi-dimensional context space is meaningful and can be exploited to select communication partners.

Experimenting with context information gathered from prototypes in our living lab environment made us again aware that privacy is a very hard problem to address in Ubiquitous Computing. Making context information anonymous is no solution to the problem. Temporal and spatial correlation of different bits and pieces of context information is very powerful. Often the information that can be inferred from correlating parts reveals more than the sum of the original pieces. Building privacy into devices and architectures at a low system level seems challenging but could offer a high degree of efficiency.

9.3 Concluding remarks

Despite Ubiquitous Computing entering the teaching curriculum and many institutions conducting research in this area, it is still a field in its infancy. It appears that slowly a common and shared understanding of what "good practice" and "valid research" in Ubiquitous Computing means is evolving.

Many advances in technologies, seen as enabling technologies for Ubiquitous Computing, are taking place. In particular in the areas of wireless networks, processing, and sensing significant progress can be observed. However many of the research demonstrators and prototypes have still a long way to go before they will be found in the "real world". For many context-aware applications it becomes apparent that they are most useful when integrated in the environment and with other systems. This is usually not a problem when building a particular prototype, however it makes the deployment in real life very hard.

In Weiser's article he characterises Ubiquitous Computing technologies that disappear as follows:

> *"They weave themselves into the fabric of everyday life until they are indistinguishable from it."* [Weiser,91]

This is certainly true from a psychological point of view. However to make Ubiquitous Computing reality it often means that:

> ***We have to weave technologies into the fabric of everyday life until they are indistinguishable from it.***

When developing concepts, models, and systems it is important to realise and keep in mind that the technologies have to be interwoven with the fabric of everyday life.

References

[Able,90] Able, K. P. & Able, M. A., “Calibration of the magnetic compass of a migratory bird by celestial rotation.” Nature, 347:378–389. 1990.

[Abowd,97] Abowd, G.D., Atkeson, C.G., Hong, J., Long, S., Kooper, R., Pinkerton, M., “Cyberguide: A Mobile Context-Aware Tour Guide”. ACM Wireless Networks 3. 421-433. 1997.

[Abowd,99] Abowd, G.D., “Classroom 2000: An Experiment with the Instrumentation of a Living Educational Environment”, IBM Systems Journal, Special issue on Pervasive Computing, Volume 38, Number 4, pp. 508-530, October 1999.
http://www.research.ibm.com/journal/sj/384/abowd.html

[Abowd,00] Abowd, G. D., Mynatt, E. D., “Charting Past, Present and Future Research in Ubiquitous Computing”, ACM Transactions on Computer-Human Interaction, Special issue on HCI in the new Millenium, 7(1):29-58, March 2000.

[Abowd,01] Abowd, G.D.; Brumitt, B.; Shafer, S. (Eds.): “Ubicomp 2001: Ubiquitous Computing”, Proceedings of the 3rd International Conference Ubiquitous Computing, Lecture notes in computer science; Vol 2201. Springer-Verlag., September 2001.

[Addlesee,97] Addlesee, M.D., Jones, A., Livesey, F., and Samaria, F.: ORL Active Floor. In IEEE Personal Communications, Vol.4, No 5, pp. 35-41, October 1997.

[Addlesee,01] Addlesee, M., Curwen, R., Hodges, S., Newman, J., Steggles, P., Ward, A., Hopper, A., “Implementing a Sentient Computing System”, Cover Feature in IEEE Computer, Vol. 34, No. 8, pp 50-56, August 2001.
http://www.uk.research.att.com/pub/docs/att/tr.2001.8.pdf

[Affective,02] Affective Computing Research at MIT, “Research on Sensing Human Affect” 2002, http://affect.media.mit.edu/AC_research/sensing.html

[Aleksander,95] Aleksander, I. and H. Morton. An introduction to neural computing (2 ed.). London, U.K.: Chapman and Hall. 1995

[Alexander,77] Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S., “A Pattern Language: Towns, Buildings, Construction”, Oxford University Press, New York. 1977.

[Alexander,79] Alexander, C. “The Timeless Way of Building”, New York: Oxford University Press, 1979.

[Analog,01] Analog Devices, “ADXL202 Datasheet”, 2001. http://products.analog.com/products/info.asp?product=ADXL202

[Antifakos,02] Antifakos, S., Michahelles, F., Schiele, B., “Proactive Instructions for Furniture Assembly”, UBICOMP 2002, International Conference on Ubiquitous Computing, Goteborg, Sweden, September 2002.

[Aristotle,00] Aristotle, “On the Soul”, (de anima), Translated by J. A. Smith, 2000. http://classics.mit.edu/Aristotle/soul.html

[AT&T,01] AT&T Laboratories Cambridge, “Sentient Computing Project Home Page.” 2001. http://www.uk.research.att.com/spirit/.

[Baeza-Yates,99] Baeza-Yates R. and Ribeiro-Neto B., "Modern Information Retrieval", Addison Wesley, New York, 1999.

[Bahl,00] Bahl, P., Padmanabhan, V., “RADAR: An in-building RFbased user location and tracking system“, In Proceedings of IEEE INFOCOM, volume 2, pages 775--784, March 2000.

[Baltes,01] Baltes, H., Gvpel, W., Hesse, J., “Sensors, Sensors Update 9“, Wiley, September 2001.

[Beadle,97] Beadle, H.W.P., Maguire, G.Q. and Smith, M.T. Smart Badge: It beeps, It flashes, It knows when you are hot and sweaty. IEEE Intl. Symposium on Wearable Computing, Cambridge, MA, USA, Oct. 1997.

[Beigl,98] Beigl, M. Schmidt, A., Lauff, M., Gellersen, H.W., "The UbicompBrowser", 4th ERCIM Workshop on "User Interfaces for All", Sweden, 19-21 October 1998.

[Beigl,00] Beigl, M. "Kommunikation in interaktiven Räumen", Dissertation (PhD Thesis), Karlsruhe University, October 2000.

[Beigl,01] Beigl, M., Gellersen, H-W., Schmidt, A.: MediaCups: Experience with Design and Use of Computer-Augmented Everyday Objects, Computer Networks, Special Issue on Pervasive Computing, Elsevier, Vol. 35, No. 4, Elsevier, p. 401-409, March 2001.

[Beigl,02] Beigl, Michael, "MediaCup Project Page", 2002. http://mediacup.teco.edu

[Bergman,00] Bergman, E., (Ed.), "Information Appliances and Beyond", Morgan Kaufmann Publishers, February, 2000.

[Borchers,01] Borchers, J., "A Pattern Approach to Interaction Design", John Wiley & Sons; 2001.

[Borriello,02] Borriello, G.; Holmquist, L.E. (Eds.): "Ubicomp 2002: Ubiquitous Computing", Proceedings of the 4th International Conference on Ubiquitous Computing, Lecture notes in computer science; Vol 2498. Springer-Verlag., September 2002.

[Brooks,98] Brooks, R.R., Iyengar, S.S., "Multi-Sensor Fusion". Prentice Hall, 1998.

[Brown,96] Brown, P.J., "The Stick-e Document: A Framework For Creating Context-aware Applications", In the Proceedings of the Electronic Publishing, pp. 259-272, Laxenburg, Austria, IFIP. September 1996.

[Brown,97] Brown, P.J., Bovey, J. D. and Chen, X., "Context-Aware Applications: From the Laboratory to the Marketplace", IEEE Personal Communications, 4(5): pages 58-64, October 1997.

[Brown,98b] Brown, P.J., "Triggering information by context", Springer-Verlag, Personal Technologies 2(1), : pp. 1-9. September 1998.

[Brumitt,00] Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S., "EasyLiving: Technologies for Intelligent Environments", In Hans-W Gellersen and P. Thomas, (Eds), HUC2000, Second International Symposium on Handheld and Ubiquitous Computing,

Bristol, UK, September 25-27, 2000. Springer-Verlag as Lecture Notes in Computer Science, vol. 1927, pp. 12-29. September 2000.

[Brumitt,00a] Brumitt, B., Krumm, J., Meyers, B., and Shafer, S.: "Ubiquitous Computing and the Role of Geometry". In IEEE Personal Communications, Vol 7, No. 5, pp. 41-43. October, 2000.

[Bulusu,00] Bulusu, N., Heidemann, J., Estrin, D., "GPS-less Low Cost Outdoor Localization For Very Small Devices", IEEE Personal Communications, Special Issue on Smart Spaces and Environments, Vol. 7, No. 5, pp. 28-34, October 2000.

[Burkhardt,01] Burkhardt, J. (Ed.), Henn, H., Hepper, S., Rindtorff, K., Schaeck, T. "Pervasive Computing: Technology and Architecture of Mobile Internet Applications", Addison-Wesley, November 2001.

[Butz,00] Butz, A., Baus, J., Krüger, A., "Augmenting Buildings with Infrared Information", Proceedings of the International Symposium on Augmented Reality (ISAR), IEEE Computer Society Press, 2000.

[Buxton,97] Buxton, W., "Living in Augmented Reality: Ubiquitous Media and Reactive Environments" K. Finn, A. Sellen & S. Wilber (Eds.). Video Mediated Communication. Hillsdale, N.J.: Erlbaum, 363-384, 1997.

[Cakmakci,02] Cakmakci, O., Coutaz, J., Van Laerhoven, K., Gellersen, H.-W., "Context Awareness in Systems with Limited Resources". In Proc. of the third workshop on Artificial Intelligence in Mobile Systems (AIMS), pp. 21-29, ECAI 2002, Lyon, France. 2002.

[Callaghan,01] Callaghan V, Clarke G, Colley M, Hagras H "Embedding Intelligence: Research Issues for Ubiquitous Computing", The 1st Equator IRC Workshop on Ubiquitous Computing, 13-14 Sept 2001, Nottingham UK. 2001.

[CCS,02] CCS, Inc., "CCS PCM compiler for PIC MCU", 2002. http://www.ccsinfo.com/

[Chen,99] Chen, D., Schmidt, A., Gellesen, H.W., "An Architecture for Multi-Sensor Fusion in Mobile Environments", Proceedings International Conference on Information Fusion, Sunnyvale, CA, USA, volume II, pp 861-868, July 1999.

[Cheverst,98] Cheverst, K., Davies, N., Mitchell, K., Friday, A., "Design of an Object Model for a Context-Sensitive Tourist Guide", Proceedings of the IMC'98 Workshop on Interactive Applications of Mobile Computing, Rostock, Germany, November 1998.

[Cheverst,00] Cheverst K., Davies N., Mitchell K., Friday A. and Efstratiou C., "Developing Context-Aware Electronic Tourist Guide: Some Issues and Experiences", Proceedings of CHI'2000, Netherlands, pp. 17-24, April 2000.

[Cheverst,00a] Cheverst K., Davies N., Mitchell K. & Friday A., "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project", Proceedings of MOBICOM'2000, Boston, ACM Press, pp. 20-31, August 2000.

[Cheverst,01] Cheverst, K., Davies, N., Mitchell, K. and Efstratiou, C., "Using Context as a Crystal Ball: Rewards and Pitfalls", Personal Technologies Journal, Vol. 3 No5, pp. 8-11 2001.

[Cheverst,01a] Cheverst K., Davies N., Mitchell K. and Smith P., "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.

[Colorado,01]. "The Adaptive House" Boulder, Colorado, 2001.
http://www.cs.colorado.edu/~mozer/nnh/

[Crabtree,01] Crabtree, A., "Wild Sociology: Ethnography and Design", Ph.D. Thesis, Lancaster University: Sociology Department. 2001.

[Crabtree,01a] Crabtree, A., Hemmings, T., Rodden, T. and Schnädelbach, H., "Patterns of Technology Usage in the Home: Domestic Legacy and Design", Technical Report Equator-01-015, 2001.

[Crowley,02] Crowley, J. L., Coutaz, J., Rey, G., Reignier, P., "Perceptual Components for Context Aware Computing", UBICOMP 2002, International Conference on Ubiquitous Computing, Goteborg, Sweden, September 2002.

[Curwen,99] Curwen, R., Hopper, A., Steggles, P. Ward, A., "Sentient Computing", AT&T Laboratories Cambridge Technical Report 1999.13 (video), 1999.
http://www.uk.research.att.com/pub/videos/qsif-200/spirit-qsif-200.mpg

[Dahlbäck,93] Dahlbäck, N., Jönsson, A., & Salber, D. " Wizard of Oz Studies - Why and How?" In Gray, W.D., Hefley, W.E., and Murray, D. (Eds) Proceedings of the 1993 ACM International Workshop on Intelligent User Interfaces, pp 193-200. ACM Press: New York. 1993.

[Darwin,59] Darwin, C., "On the Origin of Species", John Murray, 1859. Online Version at http://www.literature.org/authors/darwin-charles/the-origin-of-species/.

[Davies,98] Davies, N., Mitchell, K., Cheverst, K. and Blair, G.S., "Developing a Context Sensitive Tourist Guide", Proc First Workshop on Human Computer Interaction for Mobile Devices, Glasgow. March 1998.

[Davies,01] Davies, N., Cheverst, K., Mitchell, K. and Efrat, A., "Using and Determining Location in a Context-Sensitive Tour Guide", IEEE Computer Journal, Vol. 34, No. 8, pp. 35-41, August 2001.

[Davies,02] Davies, N., Gellersen, H.W. "Beyond Prototypes: Challenges in Deploying Ubiquitous Systems", IEEE Pervasive Computing Vol 1 No 1, March 2002.

[DC,02] "The Disappearing Computer Initiative", proactive initiative of the Future and Emerging Technologies (FET) activity of the Information Society Technologies (IST) research program. European commission, 2002. http://www.disappearing-computer.net/

[Dey,98] Dey, A.K., Abowd, G.D. and Wood, A., "CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services", Knowledge Based Systems 11(1): pp. 3-13. September 30, 1998.

[Dey,99] Dey, A.K., Salber, D., Abowd, G.D., "A Context-based Infrastructure for Smart Environments", In the Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99), pp. 114-128, Dublin, Ireland, Springer Verlag. December 13-14, 1999.

[Dey,00] Dey, A., "Providing Architectural Support for Building Context-Aware Applications", Ph. D. Thesis Dissertation, College of Computing, Georgia Tech, December 2000.

[Dey,02] Dey, A. "Evaluation of Ubicomp Applications and Systems", Summer School on Ubiquitous and Pervasive Computing, Schloss Dagstuhl, Germany, August 7-14, 2002. http://www.inf.ethz.ch/vs/events/dag2002/program/lectures/dey_3.pdf

[eGadget,02] "e-Gadgets Project", European Disappearing Computer project. 2002. http://www.extrovert-gadgets.net

[Dix,98] Dix, A., Finlay, J., Abowd, G., and Beale, R. Human Computer Interaction, 2 a Ed., Prentice Hall Europe, 1998.

[Dix,02] Dix, A.. "Beyond intention - pushing boundaries with incidental interaction." Proceedings of Building Bridges: Interdisciplinary Context-Sensitive Computing, Glasgow University, 9 Sept 2002.

[Engelbart,62] Engelbart, D. C., "Augmenting Human Intellect: A Conceptual Framework", Summary Report, AFOSR-3233, SRI Project No. 3578, October 1962. http://www.bootstrap.org/augment/AUGMENT/133182-0.html

[EQUATOR,02] "The EQUATOR Interdisciplinary Research Collaboration", 2002. http://www.equator.ac.uk

[Essa,99] Essa, I., Abowd, G.D., "Building an Aware Home: Understanding the symbiosis between computing and everyday activities.", Presentation given at MERL, Georgia Institute of Technology, 1999. http://www.cc.gatech.edu/fce/house/presentations/merl99/.

[Estrin,02] Estrin, D., Culler, D.,Pister, K., Sukjatme, G., "Connecting the Physical World with Pervasive Networks", IEEE Pervasive Computing, 1(1):59-69, Jan. 2002.

[ETH,02] Distributed Systems Group, ETH Zurich, "The Smart-Its Project", 2002. http://www.inf.ethz.ch/vs/res/proj/smartits.html

[Fausett,94] Fausett, L., "Fundamentals of Neural Networks: Architectures, Algorithms, and Applications", Prentice-Hall, 1994.

[Figaro,02] Figaro, "Semiconductor Gas Sensor Technology", 2002. http://www.figarosensor.com/

[Fuller,00] Fuller, G.F., "Falls in the Elderly", American Academy of Family Physicians, April 1, 2000. http://www.aafp.org/afp/20000401/2159.html

[Gamma,95] Gamma, E. R., Helm, R., Johnson, R., and Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[Gatech,00] "The Aware Home Research Initiative". Georgia Institut of Technology, 2000. http://www.cc.gatech.edu/fce/ahri/.

[Gelernter,82] Gelernter, D. and Bernstein, A., "Distributed Communications via Global Buffer". ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing, Ottawa, Canada, pp. 10-18, August 1982.

[Gellersen,99] Gellersen, H.W., "Handheld and Ubiquitous Computing", Processing of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC '99), Lecture notes in computer science; Vol 1707. Springer-Verlag. September 1999.

[Gellersen,99a] Gellersen, H.W, Schmidt, A., Beigl, M., "Ambient Media for Peripheral Information Display", Personal Technologies Volume 3(4), pp199-208, December 1999.

[Gellersen,99b] Gellersen, H.W., Beigl, M., "Ambient Telepresence: Colleague Awareness in Smart Environments", 1. Intl. Workshop on Managing Interactions in Smart Environments (MANSE 99), Dublin, Irland, Dec 1999 & Springer Verlag: Managing Interactions in Smart Environments, P.Nixon, G.Lacey, S.Dobson ed, pp 80-88, 1999.

[Gellersen,00] Gellersen, H-W., Beigl, M., Schmidt. A, "Sensor-based Context-Awareness for Situated Computing", Workshop on Software Engineering for Wearable and Pervasive Computing SEWPC00 at the 22nd Int. Conference on Software Engineering ICSE 2000. Limerick, Ireland, 6.June 2000.

[Gellersen,02] Gellersen, H-W., Schmidt, A., Beigl, M. "Multi-Sensor Context-Awareness in Mobile Devices and Smart Artefacts", ACM journal Mobile Networks and Applications (MONET), Vol. 7, No. 5. October 2002.

[Gellersen,02a] Gellersen, H.W., Schmidt, A., "Look who's visiting: supporting visitor awareness in the web", International Journal of Human Computer Studies IJHCS 56(1), pp. 25-46, January 2002.

[Golding,99] Golding, A. R. and Lesh, N. "Indoor navigation using a diverse set of cheap, wearable sensors", Proceedings of the third International Symposium on Wearable Computers. San Francisco, California, pp. 29-36, 18 - 19 October, 1999.

[Goldstein,97] Goldstein, E.B., "Wahrnehmungspsychologie. Eine Einführung", Spektrum Akad. Vlg., Hdg., 1997.

[Greene,02] Greene, K., "Support for nursing-home cameras", THE WALL STREET JOURNAL, March 7, 2002, http://export.msnbc.com/news/720532.asp?cp1=1

[GUIDE,01] "The GUIDE Project Home Page", Lancaster University, 2001. http://www.guide.lancs.ac.uk/overview.html

[Göpel,95] Göpel, W., Hesse, J, Zemel, J.N., "Sensors, Volume 9, Sensors Volume 9: Cumulative Index: A Comprehensive Survey", Wiley, 1995.

[Hagras,02] Hagras, H. A. K., Callaghan, V., Clarke, G. S., Colley, M. J., Pounds-Cornish, A., Holmes, A., and Duman, H., "Incremental Synchronous Learning for Embedded-Agents Operating in Ubiquitous Computing Environments", Soft Computing Agents: A New Perspective for Dynamic Information Systems, IOS Press, 2002.

[Handel,89] Handel, S. Listening: An Introduction to the Perception of Auditory Events. MIT Press, 1989.

[Harter,94] Harter, A. and Hopper, A. "A Distributed Location System for the Active Office.", IEEE Network, Vol. 8, No. 1, 1994.

[Healey,98] Healey, J. and Picard, R. StartleCam: A Cybernetic Wearable Camera. Proceedings of the International Symposium on Wearable Computing, pp. 42-49, Pittsburgh, Pennsylvania, 19-20 October 1998.

[Hinckley,99] Hinckley, K., Sinclair, M., "Touch-Sensing Input Devices", ", Proceedings of the Conference on Human Factors in Computing Systems (CHI '99), Pittsburgh, PA, May 15-20, pp. 223-230, 1999.

[Hofmann,97] Hofmann-Wellenhof, B., Lichtenegger, H., Collins, J. "Global Positioning System: Theory and Practice", Springer Verlag, 4th Rev edition, May 1997.

[Höllerer,99] Höllerer, T., Feiner, S., Terauchi, T., Rashid, G., Hallaway, D., "Exploring MARS: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System", Computers and Graphics, 23(6), pp. 779-785, Elsevier Publishers, Dec. 1999.

[Honeywell,02] Honeywell International, "Magnetic Sensors, Datasheets", 2002. http://www.ssec.honeywell.com/magnetic/datasheets.html

[Hopper,99] Hopper, A., "Sentient Computing", The Clifford Paterson Lecture, Phil. Trans. R. Soc. Lond. A (2000) 358, 2349-2358, 1999.
http://www.uk.research.att.com/pub/docs/att/tr.1999.12.pdf

[Hull,97] Hull, R.; Neaves, P.; Bedford-Roberts, J., "Towards Situated Computing", Tech Reports: HPL-97-66, HP Labs Bristol, 1997.

[Intille,02] Intille, S.S., "Change Blind Information Display for Ubiquitous Computing Environments", UBICOMP 2002, International Conference on Ubiquitous Computing, Goteborg, Sweden, September 2002.

[Jayant,99] Jayant, N., Abowd, G., Jayaraman, S., Ingram, M.A., "Enhancing the Quality of Life with Context-Aware Computing, Personalized Information Processing and Secure Broadband Communications", NSF 99-167 PREPROPSAL: ITR-RC+ HCI, 1999. http://www.cc.gatech.edu/fce/house/proposals/final-pre-proposal.doc.

[Kidd,99] Kidd, C. D., Orr, R. J., Abowd, G. D., Atkeson, C. G., Essa, I. A., MacIntyre, B., Mynatt, E., Starner, T. E. and Newstetter, W., "The Aware Home: A Living Laboratory for Ubiquitous Computing Research", Proc. of the Second International Workshop on Cooperative Buildings (CoBuild'99), October 1999.

[Kirsh,95] Kirsh, D. "The Intelligent Use of Space", Journal of Artificial Intelligence, 73 (1-2), 31-68, 1995.
http://icl-server.ucsd.edu/~kirsh/Articles/Space/AIJ1.html

[Kistler,02] Kistler force plate, 2002. http://www.kistler.com/

[Konomi,99] Konomi, S., Müller-Tomfelde, C., Streitz, N., "Passage: Physical Transportation of Digital Information in Cooperative Buildings", In: Streitz, N., Siegel, J., Hartkopf, V., Konomi, S. (Eds.), Cooperative Buildings - Integrating Information, Organizations, and Architecture. Proceedings of the Second International Workshop (CoBuild'99). LNCS 1670. pp. 45 -54. Springer: Heidelberg. 1999.

[Krüger,01] Krüger, G., Schmidt, A., "Web Engineering", Course at University of Karlsruhe, 2001. http://www.teco.edu/lehre/webe/

[Kymissis,98] Kymissis, J., Kendall, C., Paradiso, J., Gershenfeld, N,. “Parasitic power harvesting in shoes”. In Proc. of the Second IEEE International Conference on Wearable Computing (ISWC), IEEE Computer Society Press, pages pp. 132-139, October 1998.

[Laerhoven,99] Van Laerhoven, K. “Online Adaptive Context Awareness, starting with low-level sensors”. Thesis at the Free University of Brussels (VUB). Brussels, Belgium.1999.

[Laerhoven,00] Van Laerhoven, K., Cakmakci, O., “What shall we teach our pants?”, Proc. of the Fourth Internation Symposium on Wearable Computers, ISWC 2000, Atlanta, 2000.

[Laerhoven,02] Van Laerhoven, K., Schmidt, A., Gellersen, H.W., "Multi-Sensor Context-Aware Clothing". In Proceedings of the sixth International Symposium on Wearable Computers, ISWC 2002, Seattle, WA. IEEE Press. 2002.

[LaMarca,02] LaMarca, A., Brunette, W., Koizumi, D., Lease, M., Sigurdsson, S.B., Sikorski, K., Fox, D., Borriello, G., “Making Sensor Networks Practical with Robots”, International Conference on Pervasive Computing (Pervasive 2002), p. 152 ff., Zurich, Switzerland, September 2002.

[Lenat,98] Lenat, D. “The Dimensions of Context Space.”, Technical report, CYCorp, October 1998. Invited talk at the conference Context 99.
http://www.cyc.com/context-space.rtf.

[Leonhardt,96] Leonhardt, U., Magee, J., Dias, P. “Location Service in Mobile Computing Environments.” Computer & Graphics. Special Issue on Mobile Computing. Volume 20, Numer 5, September/October 1996.

[Leonhardt,98] Leonhardt, U. "Supporting Location-Awareness in Open Distributed Systems". Phd Thesis, Department of Computing, Imperial College of Science, University of London. 1998. http://www-dse.doc.ic.ac.uk/~ul/pdf/thesis_w_bookmarks.pdf

[Letham,01] Letham, L., “GPS Made Easy : Using Global Positioning Systems in the Outdoors”, Mountaineers Books, 3rd Edition, February 2001.

[Lieberman,00] Lieberman, H., Selker, T., “Out of Context: Computer Systems That Adapt To, and Learn From, Context”, IBM Systems Journal, Vol 39, Nos 3&4, pp. 617-631, 2000.

[Long,96] Long, S., Kooper, R., "Rapid Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study", Proc. 2nd ACM International Conference on Mobile Computing (MobiCom), pp 97-107, Rye, ACM Press, New York, 1996.

[Mankoff,01] Mankoff, J., "An architecture and interaction techniques for handling ambiguity in recognition-based input", Ph. D. Thesis Dissertation, College of Computing, Georgia Tech, May 2001.

[Mann,01] Mann, S., Niedzviecki, H., "Digital Destiny and Human Possibility in the Age of the Wearable Computer", Doubleday of Canada, November 2001.

[Maulsby,93] Maulsby, D., Greenberg, D. & Mander, R., "Prototyping an Intelligent Agent through Wizard of Oz", Proceedings of InterCHI `93, 277-284, April 24-29, Amsterdam. 1993.

[Microchip,02] Microchip Technology Inc., "PIC Microcontroller", 2002. http://www.microchip.com/

[Microsoft,00] "The Easy Living Project", 2000. http://research.microsoft.com/easyliving/

[Miller,68] Miller, R. B. "Response time in man-computer conversational transactions", Proc. AFIPS Fall Joint Computer Conference Vol. 33, pp. 267-277, 1968.

[MIT,02] MIT Project Oxygen. "Pervasive Human-Centred Computing". MIT Laboratory for Computer Science, July 2002. http://oxygen.lcs.mit.edu/Overview.html

[Mitchell,02] Mitchell, K, "Supporting the Development of Mobile Context-Aware Computing", Ph.D. Thesis, Department of Computing, Lancaster University, January 2002.

[Moberg,02] Moberg Research, Inc., "BrainBall", 2002. http://www.moberg.com/MobergFiles/products/entertainment/brainball.htm

[Mozer,98] Mozer, M. C., Miller, D., "Parsing the stream of time: The value of event-based segmentation in a complex, real-world control problem", C. L. Giles & M. Gori (Eds.), Adaptive processing of temporal information (pp. 370-388). Berlin: Springer Verlag. 1998.

[Mozer,99] Mozer, M. C., "An intelligent environment must be adaptive", IEEE Intelligent Systems and their Applications, 14(2) , 11-13, 1999.

[Murata,99] Murata Manufacturing Ltd., “Gyrostar – Piezoelectric vibrating gyroscope”, 1999. http://www.murata.com/catalog/s42e2.pdf

[Nakanishi,00] Nakanishi, Y., Tsuji, T., Ohyama, M., Hakozaki, K., “Context Aware Messaging Service: a Dynamical Messaging Delivery using Location Information and Schedule Information”, Journal of Personal Technologies, Vol.4, No.4, pp.221-224, 2000.

[National,02] National Semiconductor Corporation, “LM3915, Dot/Bar Display Driver”, 2002. http://www.national.com/ds/LM/LM3915.pdf

[Navas,97] Navas, J.C., Imielinski, T., “Geographic Addressing and Routing”, Proc. of the Third ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'97), Budapest, Hungary, September 1997.

[NCHS,02] National Center for Health Statistics, 2002. http://www.cdc.gov/nchs/

[Neisser,76] Neisser, U., “Cognition and Reality”, San Francisco: Freeman, 1976.

[Newman,01] Newman, M., Edwards, K., Sedivy, J., “Building the Ubiquitous Computing User Experience”, CHI 2001 Workshop, http://www2.parc.com/csl/projects/ubicomp-workshop/, April 2001.

[Nielsen,94] Nielsen, J., “Usability Engineering”, Morgan Kaufmann, San Francisco, 1994. Excerpt on “Response Times: The Three Important Limits”, http://www.useit.com/papers/responsetime.html

[Nissanka,00] Nissanka B. Priyantha, N.B., Chakraborty, A., Balakrishnan, H., “The cricket location-support system”, In Proceedings of MOBICOM 2000, ACM Press, Boston, MA, pp 32-43, August 2000.

[Nonin,02] Nonin Medical Inc., “Xpod Pulse Oxymetry”, 2002. http://www.nonin.com/xpod.html

[Norman,98] Norman, A. D., “The Invisible Computer”, Cambridge, Massachusetts; MIT Press. 1998.

[Nose,02] The Ninth International Symposium on Olfaction and Electronic Nose - ISOEN 02. 2002. http://pendragon.eln.uniroma2.it/

[Orr,00] Orr, R. J., Abowd, G. D., "The Smart Floor: A Mechanism for Natural User Identification and Tracking", Proceedings of CHI'2000, Netherlands, (April 2000), The Hague, Netherlands, April 1-6, 2000.

[Paradiso,00] Paradiso, J.A., Hsiao, K.-Y., and Benbasat, A., "Interfacing the foot: Apparatus and applications", In Proceedings of the ACM CHI Conference. Extended Abstracts, pages 175-176. 2000.

[Park,02] Park, A.; Lipperts, S.; Wilhelm, M., "Location Based Services for Context Awareness - Moving from GSM to UMTS", In: SSGRR 2002w, L'Aquira, Italy, January 2002. http://www.ssgrr.it/en/ssgrr2002w/papers/143.pdf

[Pascoe,98] Pascoe, J., "Adding Generic Contextual Capabilities to Wearable Computers", In the Proceedings of the 2nd IEEE International Symposium on Wearable Computers (ISWC'98), pp. 92-99, Pittsburgh, PA, IEEE. October 19-20, 1998.

[Pascoe,98a] Pascoe, J, Ryan, N. S. and Morse, D. R., "Human-Computer-Giraffe Interaction - HCI in the Field", In the Workshop on Human Computer Interaction with Mobile Devices, Glasgow, Scotland. May 21-23, 1998.

[Pascoe,99] Pascoe, J., Ryan, N., "Mobile Computing in Fieldwork Environments Homepage", University of Kent, UK, 1999.
http://www.cs.ukc.ac.uk/people/staff/nsr/mobicomp/Fieldwork/index.html

[Pascoe,01] Pascoe, J. "Context-Aware Software", PhD thesis, Computing Laboratory, University of Kent at Canterbury, August 2001.

[Patent,01] Method and apparatus for providing context-based call transfer operation. US patent, US 2001/0031633 A1, Oct. 18, 2001.

[Patent,02] A system and method for supporting aware goods. International patent, WO 02/46973 A2, 13 June 2002.

[Pham,00] Pham, T.L., Schneider, G., Goose, S., Pizano, A., "Composite Device Computing Environment: A Framework for Augmenting the PDA Using Surrounding Resources", Workshop on Situated Interaction in Ubiquitous Computing at CHI2000, April 2000.
http://www.teco.edu/chi2000ws/papers/35_pham.pdf

[Phillips,02] Phillips, “KMZ51; Magnetic field sensor”, 2002. http://www-us.semiconductors.philips.com/pip/KMZ51

[Picard,97] Picard, R., Healey, J., “Affective wearables”, Personal Technologies, vol. 1, no. 4, pp. 231--240, 1997.

[Portolano,02] “Portolano: An Expedition into Invisible Computing”, University of Washington, 2002. http://portolano.cs.washington.edu/

[Portolano,99] “Portolano/Workscape: Charting the new territory of invisible computing for knowledge work”, A proposal to DARPA in response to BAA 99-07, 1999.

[Radiometrix,02] Radiometrix Ltd., “433MHz High Speed FM Radio Transceiver Module”, 2002. http://www.radiometrix.co.uk/products/bim2.htm

[Ramtron,02] Ramtron Intl. Corp., “FRAM Datasheets”, 2002. http://www.ramtron.com/products/datasheets.htm

[Randell,00] Randell, C., Muller, H., “Context Awareness by Analyzing Accelerometer Data”, Fourth International Symposium on Wearable Computers (ISWC'00), p. 175-176, Atlanta, Georgia, October 18 - 21, 2000.

[Regenstein,01] Regenstein, K. “Lokationsbestimmung für rechnerunterstützte Artefakte des Alltäglichen Gebrauchs“, Diplomarbeit (Master Thesis), Department of Physics, University of Karlsruhe, 2001.

[Rekimoto,01] Rekimoto, J., “GestureWrist and GesturePad: Unobtrusive Wearable Interaction Devices”, Proceedings of the Fifth International Symposium on Wearable Computers (ISWC'01), 2001.

[Russell,95] Russell, S.J., Norvig, P., “Artificial Intelligence: Modern Approach”, Prentice Hall; January 1995.

[Ryan,98] Ryan, N. S., Pascoe, J., Morse, D. R., "Enhanced Reality Fieldwork: the Context-aware Archaeological Assisstant", in V. Gaffney, M. van Leusen and S. Exxon (eds.) Computer Applications in Archaeology 1997, 1998. http://www.cs.ukc.ac.uk/projects/mobicomp/Fieldwork/Papers/CAA97/ERFldwk.html

[Saffo,97] Saffo, P., Sensors: The Next Wave of InfoTech Innovation, 1997 Ten-Year Forecast, Institute for the Future, 1997. http://www.saffo.com/sensors.html

[Salber,93] Salber, D., Coutaz, J. "Applying the Wizard of Oz Technique to the Study of Multimodal Systems", EWHCI'93, pp219-230. Springer-Verlag, Berlin, 1993.

[Salber,99] Salber, D., Dey, A.K. and Abowd, G.D., "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. pp 434-441, 1999.

[Sato,01] Sato, Y., Shingyouuchi, M., Furuta, T., Beppu, T., "Novel Device for Inputting Handwriting Trajectory", Ricoh Technical Report No.27, November, 2001.

[Sawhney,98] Sawhney, N., Schmandt, C., "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.

[Scheich,86] Scheich, H., Langner, G., Tidemann, C., Coles, R., Guppy, A., "Electroreception and electrolocation in platypus", Nature 319:401-404. 1986.

[Schiele,01] Bernt Schiele., B, Antifakos, S., "Beyond Position Awareness", Proceedings of the Workshop on Location Modeling at UBICOMP 2001, Atlanta, 2001.

[Schilit,94] Schilit, W. N., Adams, N. I. and Want, R., "Context-aware Computing Applications", In the Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications, pp. 85-90, Santa Cruz, CA, IEEE. December 8-9, 1994.

[Schilit,95] Schilit, W. N., "A System for Context-Aware Mobile Computing", Ph.D. Thesis, Columbia University, New York, 1995.

[Schmidt,96] Schmidt, A., "A modular neural network architecture with additional generalisation abilities for high dimensional input vectors", MSc thesis, Manchester Metropolitan University, UK, September 1996.

[Schmidt,98] Schmidt, A., Beigl, M. and Gellersen, H. W., "There is More to Context than Location", In Interactive Applications of Mobile Computing, Rostock, Germany, 24-25, November 1998.

[Schmidt,99] Schmidt, a., Beigl, M., Gellersen, H. W., "There is More to Context than Location," Computer & Graphics Journal, vol. 23, no. 6, pp. 893-902, Dec. 1999.

[Schmidt,99a] Schmidt, A., Forbess, J., "What GPS Doesn't Tell You: Determining One's Context with Low-Level Sensors", The 6th IEEE International Conference on Electronics, Circuits and Systems, September 5 - 8, 1999, Paphos, Cyprus. 1999.

[Schmidt,99b] Schmidt, A., Gellersen, H.W. and Beigl, M., "A Wearable Context-Awareness Component - Finally a Good Reason to Wear a Tie", IEEE Proceedings of the third International Symposium on Wearable Computers. pp 176-177, San Francisco, 18-19. Oct. 1999.

[Schmidt,99c] Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Laerhoven, K. V. and Velde, W. V., "Advanced Interaction in Context", In the Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC '99), pp. 89-101, Karlsruhe, Germany, Springer-Verlag. September 27-29, 1999.

[Schmidt,00] Schmidt, A., Takaluoma, A. and Mntyjrvi, J., "Context-Aware Telephony over WAP", Springer-Verlag, London, Ltd., Personal Technologies, Volume 4, pages 225-229. Short paper presented at Handheld and Ubiquitous Computing (HUC2k), HP Labs, Bristol, UK, September 2000.

[Schmidt,00a] Schmidt, A., "Implicit Human Computer Interaction Through Context", Personal Technologies Volume 4(2&3), pp191-199, June 2000.

[Schmidt,00b] Schmidt, A., Gellersen, H.W., Merz, C., "Enabling Implicit Human Computer Interaction - A Wearable RFID-Tag Reader", International Symposium on Wearable Computers (ISWC2000), pp193-194, Atlanta, GA, USA. October, 16-17, 2000.

[Schmidt,01] Schmidt, A., Gellersen, H.W.,"Modell, Architektur und Plattform für Informationssysteme mit Kontextbezug", Infromatik Forschung und Entwicklung. Band 16, Heft 4. pp213-224, November 2001.

[Schmidt,01a] Schmidt, A., Gellersen, H.W., "Visitor Awareness in the Web" 10th World-Wide Web Conference (WWW10), W3C, Hongkong, May 2001.

[Schmidt,02] Schmidt, A., Strohbach, M., van Laerhoven, K., Friday, A., Gellersen, H.W., "Context Acquisition Based on Load Sensing", UBICOMP 2002, International Conference on Ubiquitous Computing, Goteborg, Sweden, September 2002.

[Schmidt,02a] Schmidt, A., van Laerhoven, K., Strohbach, M., Gellersen, H.W., "Ubiquitous Pointing", UI4ALL 2002, 7th ERCIM Workshop User Interfaces For All, Paris, October 2002.

[Schmidt,02b] Schmidt, A., "Smart-Its technical details, schematics, PCBs", 2002. http://www.comp.lancs.ac.uk/~albrecht/smart-its/platform/

[Schmidt,02c] Schmidt, A., "Smart-Its prototyping video", DC Atelier Sep 2002. http://ubicomp.lancs.ac.uk/~albrecht/smart/

[Scholtz,01] Scholtz, J., "Workshop on Evaluation Methodologies for Ubiquitous Computing", Workshop at Ubicomp 2001. http://zing.ncsl.nist.gov/ubicomp01/

[Sensor,99] Sensor 99. 9th international trade fair and conference for sensors, transducers & systems. AMA Service. Nürnberg, Germany,1999.

[Sensor,01] Sensor 2001. Internationaler Kongress. 10. 2001 May. 1 rnberg, Oldenburg Verlag, Germany, 2001.

[Shneiderman,83] Shneiderman, B., "Direct manipulation: A step beyond programming languages", IEEE Computer, 16(8), p.57-69, August 1983.

[Siegert,96] Siegert, H.-J., Bocionek, S.: Robotik: Programmierung intelligenter Roboter. Springer 1996.

[Small,00] Small, J., Smailagic, A., Siewiorek, D., "Determining User Location For Context Aware Computing Through the Use of a Wireless LAN Infrastructure" December 2000. http://www-2.cs.cmu.edu/~aura/docdir/small00.pdf

[SMART,02] "The Smart-Its Project", European Disappearing Computer Project, 2002. http://www.smart-its.org/

[Starner,96] Starner, T., "Human-Powered Wearable Computing", IBM Systems Journal, Vol. 35, No. 3&4, pp. 618-629, 1996.

[Starner,98] Starner, T., Schiele, B., Pentland, A., "Visual Contextual Awareness in Wearable Computing." Proceeding of the Second Int. Symposium on Wearable Computing. Pittsburgh, October 1998.

[Starner,99] Starner, T. "Wearable Computing and Contextual Awareness", PhD thesis, MIT Media Laboratory, Apr 30, 1999.

[Svaizer,97] Svaizer, P., Matassoni, M., Omologo, M., "Acoustic Source Location in a Three-dimensional Space using Cross-power Spectrum Phase." Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP97), Munich, Germany, April 1997.

[TAOS,02] TAOS, Texas Advanced Optoelectronic Solutions, "Light to Volatge Familiy", 2002. http://www.taosinc.com/light_to_voltage.htm

[TEA,98] "Technology for enabling Awareness (TEA)", European Esprit Project 26900. http://www.teco.edu/tea/, 1998.

[Teco,02] TecO, University of Karlruhe, "Smart-Its Project page", 2002.
http://smart-its.teco.edu

[Thede,01] Thede, A., Schmidt, A., Merz, C., "Integration of goods delivery supervision into E-Commerce supply chain", Second International Workshop on Electronic Commerce (WELCOM'01). Heidelberg, Germany. November 16-17, 2001.

[Thomas,00] Thomas, P.; Gellersen, H.-W. (Eds.): "Handheld and Ubiquitous Computing", Proceedings of the Second International Symposium on Handheld and Ubiquitous Computing (HUC 2000), Lecture notes in computer science; Vol 1927. Springer-Verlag, September 2000.

[Traeger,94] Traeger, D.H. "Einführung in die Fuzzy – Logik", 2. Auflage. Teubner, Stuttgart. 1994.

[Vijayraghavan,01] Vijayraghavan, V., Barton, J.J., "WISE - A Simulator Toolkit for Ubiquitous Computing Scenarios", UbiTools-'01 workshop, 2001.
http://www.hpl.hp.com/personal/John_Barton/Publications/WISE_Ubitools_3.pdf

[Want,92] Want, R., Hopper, A., Falcao, V. und Gibbons, J. "The Active Badge Location System.", ACM Transcation on Information Systems 10 (1992) 1, pp. 42-47. 1992.

[Want,95] Want, R., Schilit, B.N., Adams, N.I., Gold, R., Petersen, K., Goldberg, D., Ellis J.R., Weiser, M., "The ParcTab Ubiquitous Computing Experiment", Technical Report CSL-95-01, Xerox Palo Alto Research Center, March 1995.
http://www.ubiq.com/parctab/csl9501-abstract.html

[Want,02] Want, R., Pering, T., Borriello, G., Farkas, K., "Disappearing Hardware", IEEE Pervasive Computing Journal, Vol. 1. Issue 1, pp36-47, April 2002.

[Ward,97] Ward, A. J., Hopper, A., "A New Location Technique for the Active Office", IEEE Personal Communications, vol. 4, pp. 42-47, 1997.

[Weiser,91] Weiser, M., "The Computer for the 21st Century", Scientific American, 265(3):94-104, September 1991.

[Weiser,93] Weiser, M., "Some Computer Science Issues in Ubiquitous Computing", Communications of the ACM, 36(7):75-84, 1993.

[Weiser,96] Weiser, M., "Ubiquitous computing Homepage", http://www.ubiq.com/hypertext/weiser/UbiHome.html, March 1996.

[Weiser,98] Weiser, M. & Brown, J. S. "The coming age of calm technology". In P. J. Denning & R. M. Metcalfe (Eds.), Beyond calculation: The next fifty years of computing. pp 75-85. New York, NY, 1998.
http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm

[Wejchert,00] Wejchert, J. "The Disappearing Computer", Information Document, IST Call for proposals, European Commission, Future and Emerging Technologies, February 2000. http://www.disappearing-computer.net/mission.html

[Wisneski,98] Wisneski, G., Ishii, H., Dahley, A., Gorbet, M., Brave, S., Ullmer, B. and Yarin, P. Ambient Display: Turning Architectural Space into an Interface between People and Digital Information. In Proceedings of the First International Workshop on Cooperative Buildings (CoBuild'98), Darmstadt, Germany, Springer-Verlag Heidelberg, p. 22-32, February 1998.

[Wyckoff,98] Wyckoff, P., McLaughry, S. W., Lehman T. J. and Ford, D. A., "T-Spaces", IBM Systems Journal, Vol 37, No. 3 - Java Technology. pp 454–474, 1998.

[Zadeh,73] Zadeh, L. A., "Outline of a New Approach to the Analysis of Complex Systems", IEEE Transac-tions on System, Man and Cybernetics, vol. 3, no. 1, pp. 28-44, Jan 1973.

Appendix

Appendix A: Perception

Appendix A.1: Time Domain Analysis.

To discriminate the audio signal of the contexts silence, noise, speaking, and music on a minimal computing hardware a simple time domain analysis is used. The algorithm finds the point where there is a zero crossing (red dots in Figure 30) and when a direction change happens (green dots in Figure 30). The ration between the counts is an indication of the type of audio signal.

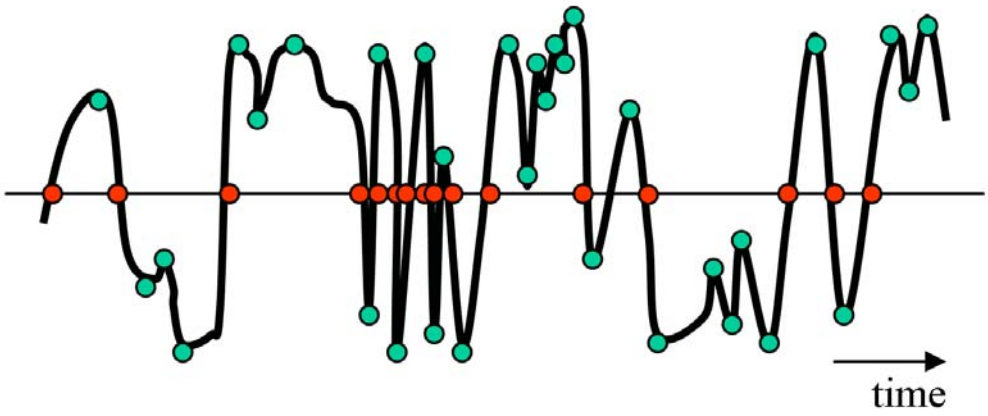


Figure 30: An example depicting zero crossings and direction changes in an audio signal.

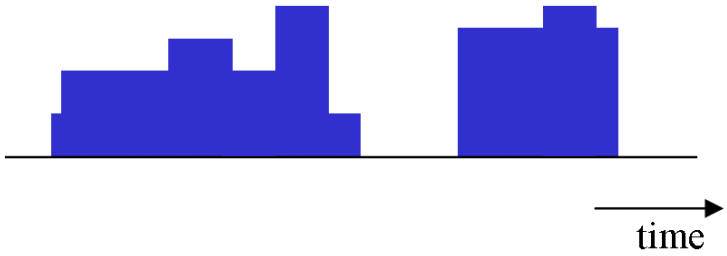


Figure 31: The graph shows the pattern that reassembles 3 seconds of audio.

If there is knowledge about the average value (`avg`) these measures can be calculated on the fly without storing all samples.

Zero crossing condition:

```
(x(t-1)>avg & x(t)<avg) | (x(t-1)<avg & x(t)>avg)
```

Direction change condition:

```
(x(t-1)>x(t-1) & x(t-1)>x(t)) | (x(t-1)<x(t-1) & x(t-1)<x(t))
```

A further algorithm that also works in the time domain indicates the level of the audio signal over time. The algorithm can be compared with what can be observed in an audio editor when a sound signal over a longer period of time is viewed. All the values are compressed in time, resulting that only the maxima are visible. In Figure 31 the resulting output is depicted. These cue can also be calculated by observing the maximum on the fly and without storing the whole data.

Appendix A.2: A Simplified Rule Set

The simplified rule set to discriminate between the situations where a device is hold in the hand, the device is stationery on the table, and the device is in a suitcase, is shown in Table 13. The recognition in this example is based on only three sensors: light, and acceleration in two directions (X and Y). The rules are built on observation of usage in contexts and from an analysis of the data collected in test scenarios. The sample data is also used to calculate the threshold values that are used for comparison:

The values `Dx` and `Dy` indicate the threshold for movement. For the data it could be observed that when the device is held by the user the standard deviation of the acceleration values always exceed a certain threshold.

The acceleration values indicate the orientation of the device when it is not moved. The collected data was used to calculate the normal values that are measured when a device is stationary on a table, denoted `Xnormal`, `Ynormal`. As sensor readings have a distribution a threshold `D` was introduced. If the difference was smaller than D it was still assumed that the position is the same.

Similarly when the device is not moved the changes in acceleration should be minimal. The quartile distance was used to check this; `Q` is the threshold that was calculated from the data sets. For light the threshold for a reading in the dark was set by the value `L`.

Appendix A.3: Recognising Events on a Surface

To detect events in the load sensing system the following algorithm is used. A 500 ms sliding window is used and only the absolute sum of load (Fx) is regarded. At the selected sampling frequency, this equates to the last 125 sample values, denoted by Fx(t),…, Fx(t-124). For each sample the cues described in Table 14 are calculated. The expressions have been selected to be as simple as possible to facilitate easy implementation on a microcontroller while at the same time yielding reliable event detection and differentiation.

Using the calculated cues three different events are discriminated by the rules described for each primitive.

```
Hand(t):-       standard_ deviation(accelX,t) > Dx,
                standard_ deviation(accelY,t) > Dy,
                % device is slightly moving in X and Y
                average(light,t)>L. % not totally dark

Table(t):-      Abs(average(accelX,t)-Xnormal)<D,
                Abs(average(accelY,t)-Ynormal)<D,
                % the device is level in X and Y
                quartile(accelX,t)<Q, quartile(accelY,t)<Q
                % the device is stationary
                average(light,t)>L. % not totally dark

Suitcase(t):-   average(light,t)<L. % it is totally dark
```

Table 13: Simplified recognition rules a context aware phone.

- **Putting an object on the surface**.

  This is characterised by an increase in the overall load. In other words, before the event the overall load is smaller than after the event ($A_s+\delta< A_e$). The threshold of weights is denoted by δ. Assuming that once the object has been put down on a surface it remains stable, it can be seen that D_s is close to zero ($D_s<\varepsilon$). In the middle of the interaction the change to the signal is greater (the moment the object hits the surface) than later (when the object is already on the surface), stated as ($D_m> D_e$).

- **Removing an object from the surface.**

  This is inverse of placing an object on the surface, so the overall load is reducing ($A_s>A_e+\delta$). To begin with the signal is stable ($D_s<\varepsilon$) and the change during the interaction is greater than at the end ($D_m> D_e$).

- **Knocking an object over.**

  When an object is knocked over, this results in a large change in the middle of the interaction, greater than at both the start and the end, and also greater than a set threshold ϕ, ($D_m>\phi \wedge D_m>D_s \wedge D_m>D_e$). As the overall weight on the surface stays the same, A_s and A_e are similar ($|A_s-A_e|<\delta$).

$A_s = \frac{\sum F_x(j)}{\frac{j=(t-124)..(t-100)}{25}}$	A_s is the average value of the first 25 values in the window that is currently processed. This is used as an indicator for the overall load on the surface before the interaction.
$A_m = \frac{\sum F_x(j)}{\frac{j=(t-101)..(t-25)}{75}}$	A_m is the average value of the middle 75 values in the window that is currently processed. This value is required for the calculation of D_m.
$A_e = \frac{\sum F_x(j)}{\frac{j=(t)..(t-24)}{25}}$	A_e is the average value of the last 25 values in the window that is currently processed. This is used as an indicator for the overall load on the surface after the interaction.
$D_s = \frac{\sum \lvert F_x(j) - A_s \rvert}{\frac{j=(t-124)..(t-100)}{25}}$	D_s is an indicator for the change in the signal during the first 25 samples. If no interaction takes place D_s is close to 0.
$D_m = \frac{\sum \lvert F_x(j) - A_m \rvert}{\frac{j=(t-101)..(t-25)}{75}}$	D_m is an indicator for the change in the signal during the middle 75 samples.
$D_e = \frac{\sum \lvert F_x(j) - A_e \rvert}{\frac{j=(t)..(t-24)}{25}}$	D_e is an indicator for the change in the signal during the last 25 samples.

Table 14: Formulae calculated to detect interaction events.

Appendix B: Load Sensing System

The load cells used in the experiments are based on resistive technology. Put simply, each cell is a wheat stone bridge providing a maximum output signal of 20mV at a driving voltage of 5V. The AD-converter included in the microcontroller can measure voltages between 0 and 5V. To best utilise this range, the output is amplified by a factor of 220 using an LM324, resulting in an output signal of 0 to 4.4V (the exact values vary slightly between the load cells). The amplified output voltage of each of the load cells is converted into a 10 bit sample; the best resolution offered by the MCU's internal AD converter. Each of the four input channels can be sampled up to 250Hz. The four input values correspond to the load that is measured on each of the load cells.

Events from the table and shelf unit are sent wirelessly using an RF transceiver module (Radiometrix BIM2) that offers data rates of up to 64kbit/s. The communication is run at 19,200 bits/s. Events, such as putting an object down at a certain position, removing an object, tracking over the surface or pressing down, are sent in a single packet. Each packet comprises a preamble, followed by a start-byte, an object identifier to determine the origin (coffee table, large table, or shelf), the event type, and the event dependent data. Finally, two bytes of 16-bit CRC are attached to ensure that the transmitted data will be received correctly. The protocol frame is depicted in Figure 33. The data acquisition unit only transmits data (there are no acknowledgements), however, in the experimental setting the protocol was very

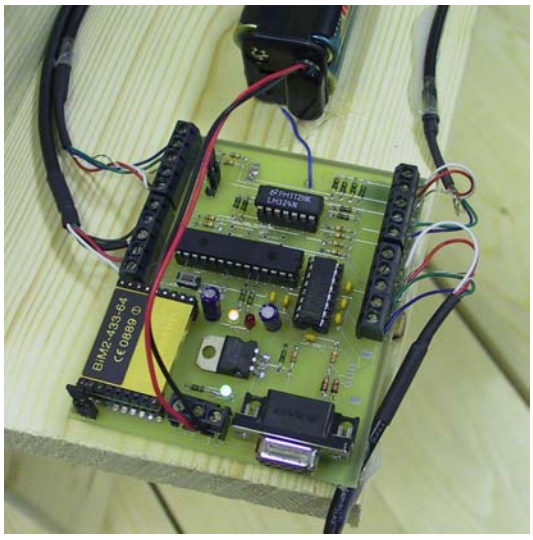


Figure 32: First generation data acquisition and communication hardware.

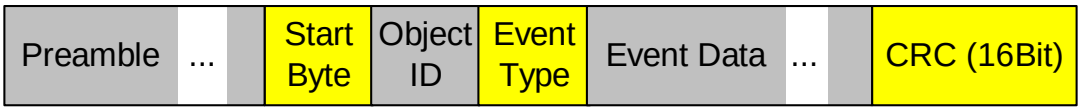


Figure 33: RF packet frame.

reliable at the low transmission speed.

The hardware module used for data acquisition (as depicted in Figure 32) also acts as a base station that receives the events and sends them to the host PC via RS-232. The floor is directly connected to the PC.

Learning from the experience gained by deploying the system a second hardware iteration that offers some improvements over our first design was designed as Add-On board to the Smart-Its platform.

Appendix C: Patterns

Appendix C.1: Context Pattern #1, battery powered hand held electronic appliance

… this pattern looks at contexts that are relevant for battery powered, hand held electronic appliances. In particular the usage of such devices and the interaction with them is at the centre of the interest to this pattern.

The **entity** is a small handheld device that incorporates electronics to provide the functionality. It is also assumed that the device is built to provide a specific functionality. Furthermore the energy for the device comes from a battery, which can also be rechargeable.

Scenario: A user takes out their mobile phone from their jacket and puts it on the desk. At lunch time they go for a walk and takes the phone in their hand and write an SMS. After finishing they put the phone in their bag. Back at the office they put the phone in the phone recharging station.

Typical **examples** of such devices are mobile phone, pager, PDAs, gaming appliances, calculators, and pocket translators.

Contexts of interest include: device in users hand, device in a pocket on the body, device lying on an open surface, device in a locker, device in a carry bag, device in operation, device picked up, device put down, device lying on a surface, and device in recharging station, light conditions, way the device is held.

The contexts mentioned above are **of interest to** the device and more specific to applications running on the device. In more general scenarios the contexts can also be of interest to anyone around.

Major constraints and **forces** on the implementation include power consumption, robustness, weight, size, and unobtrusiveness.

The prime **sensing technologies** that are appropriate include: accelerometers (position and acceleration), touch and proximity sensors on the device, light sensors at different positions of the device, temperature sensor.

As the contexts of interest are universal, on-the-fly learning is not required. Therefore the following **perception methods** are appropriate: calculation of low level cues using statistic, off-line trained supervised artificial neural networks, and rules that operate on the calculated features.

The **device architecture** for this pattern allows to generally different approaches: incorporated design and add-on design. The incorporated design is applicable when a device is newly designed. The sensing and perception technology can be closely tied in with the design, e.g. using the same processor, controller, or DSP. If context has to be provided to a device that is already designed the sensing and perception is built as a separate module that is connected to the device.

Typically **communication** will be realised in a wired way, because the sensing and perception part becomes a component of the system. If an incorporated architecture is followed the communication can happen on various points in the design, for add-on components a serial protocol, such RS232 or I2C, are useful. If the contexts should also be available in the environment, wireless communication has to be taken into account.

Implementation examples are a context-aware mobile phone [Schmidt,99c], [TEA,98] and a context-aware PDA [Schmidt,98], [Schmidt,99].

This pattern is related N.N.

Appendix C.2: Context Pattern #2, mains powered stationary appliance

… this pattern looks at contexts that are applicable to stationary appliances that are connected to a permanent power supply. In particular the interaction with these devices is at the centre of the interest to this pattern.

The **entity** is a stationary device that is based in the environment. It is electrically powered and connected to mains. It incorporates electronics to provide a specific functionality.

Scenario: A person is in the living room and switches the TV on to watch the news. The phone rings and they pick up the hand set and mute the TV. After talking on the phone they increase the volume on the TV again. After the news they switch off the TV and go to the kitchen, and open the fridge to get a drink. Then they switch on the stereo to listen to some music. When their partner comes they start to talk and one of them lowers the volume of the stereo.

Typical **examples** of such devices are lights, TV sets, radios, stereos, PCs, fixed line phones, fridges, and washing machines.

Contexts of interest include: someone is approaching the device, interaction with the device, mode of interaction (device dependent), the noise around the device, the light conditions around, and whether or not the device is in operation.

The contexts mentioned above are **of interest to** the device and the services provided by the device. However the contexts can also be of interest to other devices in the environment.

Few constraints and **forces** are seen in this pattern, main concerns are incorporation in the design, reliability, and introduced cost.

The prime **sensing technologies** that are appropriate include: audio sensors, touch and proximity sensors on the device, light sensors, and also logical sensors that allow to access information about the state of the device (e.g. which channel is selected).

Most of the contexts of interest are universal and can be set up before, so that no on-the-fly learning is required. Given the fact that there are neither power nor size restrictions any **perception method** that appears to be appropriate could be used. However to keep the introduced cost down, processing should be kept as simple as possible.

The **device architecture** applicable for this pattern also allows to different approaches: incorporated design and add-on design. For new developments building-in context-perception into the design is the most appropriate solution. For devices that are programmable and offer a connection add-on devices can be built.

If perception is built into the device **communication** will be implicit in the hardware and software. For add-an modules a connection will be most easily realised in a wired way. To provide the contexts that are acquired in the device to other entities in the system further communication is required, here power line communication or a wireless solution should be considered.

Implementation examples are a context-aware stereo and TV.

This pattern is related N.N.

Appendix C.3: Context Pattern #3, non electronic portable every day objects

… this pattern looks at contexts that are relevant for every day objects that are portable. In particular the usage and whereabouts of such objects is central to this pattern.

The **entity** is an every day object that people can carry around. The object does not necessarily provide a specific functionality. A main property of such objects is that they are virtually maintenance free (besides from dusting or cleaning them from time to time). For its own purpose there are no electronics built into the device.

Scenario: A person walks from the kitchen to living room carrying two mugs of tea. They put the mugs down on the windowsill. After a while they hand one of the mugs over to another person and lifts up their own to drink. They keep their mug in their hands while talking to the other person. From time to time they drink from the mug again. After a while they put the mug down on the coffee table.

Typical **examples** of such objects are cups, mugs, wineglasses, books, scissors, pens, and paper clips.

Contexts of interest include: object in users hand, object in a pocket or carry bag, object places on an open surface, object in a cupboard, object operated (entity dependent, e.g. drinking from a cup, reading the book, cutting with the scissors), object picked up, object put down, way the object is held, other object in the neighbourhood.

The contexts mentioned above are **of interest to** the environment and the underlying system. In particular it is also of interest to electronic artefacts that provide functionality that is related to object. It is not of interest to the object itself, because it is passive.

Major constraints and **forces** on the implementation include power consumption, unobtrusiveness, robustness, and portability. The constraints are particular crucial, because the objects are not regarded as ‘electronic gadgets’ and therefore additional maintenance (e.g. rebooting, recharging) is in most cases not acceptable for the user.

The prime **sensing technologies** that are appropriate include: zero energy sensors (e.g. ball switches to find out when an object was picked up), accelerometers (position and acceleration), touch and proximity sensors on the device, and sensors that are specifically aimed to get knowledge about properties and the operation of that object (e.g. temperature sensor in a cup to get the coffee temperature).

The contexts of these everyday objects can mainly be defined in a general way, so that no learning is required once the device is built. **Perception methods** such as calculation of low level cues using statistic, off-line trained supervised artificial neural networks, and rules that operate on the calculated features are therefore appropriate.

The **device architecture** for context-aware every day artefacts is in general an electronic add-on component where the sensors are interwoven into the physical design of the object. Here the placement of the sensors is a major concern, influencing the processing that will be needed and hence the power consumption as well as the robustness of the artefact.

Because the contexts are of interest only to the outside world **communication** from the device to its environment is required. To realise this unobtrusively and without compromising usability a wireless solution should be used.

Implementation examples are a context-aware cup [Gellersen,02] and a context-aware book. The Smart-Its project [SMART,02] takes this further and looks into how a general technical solution can be provided for this pattern.

This pattern is related N.N.

Appendix C.4: Context Pattern #4, non electronic stationary every day objects

… this pattern looks at contexts that are relevant for every day objects that are kept in the environment. In particular the interaction with such objects is central to this pattern.

The **entity** is an every day object that people keep stationery in the environment. The object does either provide a specific functionality or has a different value to the owner. These objects are most often maintenance free, besides from cleaning them. These artefacts do not include electronics to fulfil their purpose.

Scenario: A person has a number of paintings and photos in their apartment, which are framed and mounted on the walls in the kitchen and the bedroom. In the living room they have a wooden sculpture of an elephant. On the windows there are curtains and the rooms are separated by doors.

Typical **examples** of such objects are paintings, pictures frames, sculptures, art pieces, doors, and curtains.

Contexts of interest include: someone is approaching the artefact, interacting with the artefact, mode of interaction (depends on the artefact), and the surrounding conditions such as temperature, light, and noise.

The contexts are **of interest to** the environment and the underlying system, only. The context is not helpful for the artefact because the artefact is passive. The contexts however can be of particular interest to electronic devices that are related to the artefact, e.g. a spot light for a certain sculpture.

Major constraints and **forces** on the implementation include unobtrusiveness, robustness, and in some cases power consumption. Especially for art pieces and art related artefacts it is important that the expression of the artefact is not changed by making it context-aware. These artefacts are not seen as technology and hence therefore additional maintenance is hardly acceptable for the user.

The prime **sensing technologies** include touch and proximity sensors, preferably implemented by extrinsic sensing. Furthermore audio sensors and light sensors can be deployed to get access information about the environment. For objects that are fixed in the environment and that can be moved (e.g. doors, curtains) also accelerometers can give valuable information.

To some extent these contexts of these everyday artefacts are general and therefore can be built independently of the environment. **Perception methods** depend very much on the decision of type of sensors and the selected contexts that are of interest in this pattern.

For the **device architecture** two approached can be pursued. When realising sensing, extrinsically a context-component that is only logically attached to the artefact, which is designed so that it can be physically placed somewhere else. This sensing device is then designed as a standalone component. If sensing is built into the artefact sensors become physically integrated, however because there are no electronics in the device the sensing component is also stand-alone.

The context should be made available to other devices in the neighbourhood therefore external **communication** from the device to its environment is required. To realise

this in an unobtrusive way a wireless or wired solution may be used, depending on the artefact and the sensing approach taken.

Implementation examples are a picture frame that is aware of its observer, door that knows its state [Buxton,97], and a curtain that also knows it state.

This pattern is related N.N.

Appendix C.5: Context Pattern #5, non portable furniture with horizontal surfaces

… this pattern looks at contexts that are relevant for furniture. It is concerned with furniture that is fixed in the environment.

The **entity** in this pattern is a piece of furniture that is stationary in a place after it has been set up. A discriminating property is that these artefacts incorporate surfaces where other thinks can be put on. In this pattern the term furniture is extended to also cover fixed installations and decorations.

Scenario: A person opens the door, put their coat on the coat hanger and walks into the kitchen. They open a cupboard, get out a bowl, put it on the kitchen table, then open another cupboard, get out some crisps and put them into the bowl. Then they sit down on the sofa putting the bowl down next to themselves on the windowsill.

Typical **examples** of such artefacts are tables, lockers, cupboards, sofas, windowsills, and raised floors.

Contexts of interest include: someone is interacting with the artefact, something is put down on the surface at a certain position (or lifted), and the object is approached.

The contexts mentioned above are **of interest to** the system where the furniture is a part of. The contexts of which the artefact is aware off can be of interest to any object around.

Major constraints and **forces** on the implementation are unobtrusiveness and social acceptance of the technology. Further concerns are the design and also the robustness of the artefact.

The prime **sensing technologies** that are appropriate include: distributed weigh sensing using load cells (position of objects on the surface), touch and proximity sensors.

The basic contexts – such as where was an object put down on the surface, are general and require not learning capabilities. More complex scenarios, e.g. recognising a specific object on the surface does however require learning while in operation. For more complex contexts a whole variety of **perception methods** is appropriate because there a few constraints on computing power. Typically artificial neural networks, self learning systems, and systems based on rules can be deployed.

The **device architecture** for context-aware furniture includes the incorporation of sensors into the artefacts. Because the furniture is fixed in the environment and in general little space restrictions apply processing can be done somewhere in the artefact or even in the background.

The contexts should be communicated to other artefacts in the environment, therefore a form of **communication** is necessary. Depending on the type of furniture and how it is fixed in the environment wireless or wired communication may be appropriate. Because entities can be fairly large, e.g. a raised floor, communication between individual sensors and the processing unit may be required.

Implementation examples are a context-aware tracking table [Schmidt,02a], a weight-aware shelf, and a weight tracking floor [Addlesee,97], [Schmidt,02].

This pattern is related N.N.

Appendix C.6: Context Pattern #6, furniture on that people sit

… this pattern looks at contexts that are relevant for furniture that people primarily use to sit on.

The **entity** in this pattern is a piece of furniture that is used as seat. It can be either fixed in a certain location or portable. Furniture to sit on incorporates artefacts that are built to sit individual people or groups of people.

Scenario: A person sits down on a chair next to the kitchen table having both feet put to the floor. After a while they pick up a newspaper and lean back putting one leg over the other. When finished with reading they put the paper to the table, put both feet to the floor and get up. For watching TV they join their friend on the sofa, sitting down next to them.

Typical **examples** of such artefacts are chairs, benches, seats, sofas, and armchairs.

Contexts of interest include: someone is approaching the artefact, someone is sitting down on it, someone is getting up, number of people sitting on it, type or identity of person sitting on an entity, the way people sit on it (e.g. leaning back, dependent on the artefact), and if the artefact is moved.

The contexts mentioned above are **of interest to** the system or the environment where the furniture is a part of. The contexts of which the artefact is aware off can be of interest to any object around.

Similar to other furniture major constraints and **forces** on the implementation are unobtrusiveness and social acceptance of the technology. Further concerns are the design and also the robustness of the artefact. In case of portable furniture energy becomes an issue, too.

The prime **sensing technologies** that are appropriate include: surface weight sensing based on force sensitive resistors, distributed weight sensing in frame using load cells, acceleration sensors for portable artefacts, touch and proximity sensors.

In the single artefacts the basic contexts, such as someone is sitting on it, are general and can be implemented without online learning capabilities. Personalised contexts, e.g. the identity of the person sitting on a chair, require learning or at least calibration capabilities. The basic perception can be realised by calculating the weight, and the distribution of weight for the load sensing sensors. Statistics are suitable to calculate general features from the other sensors. For higher level complex contexts a variety of **perception methods** can be used.

The **device architecture** for context-aware seats includes the incorporation of sensors into the artefacts and into the seat surface. Processing can also be incorporated in the entity, because usually space is available.

As contexts need to be communicated to other artefacts in the environment a form of inter-device **communication** is necessary. Depending on the type of furniture and how it is fixed in the environment wireless or wired communication may be appropriate.

Implementation examples are a context-aware swivel chair, context aware sofa [Microsoft,00], and an aware bench.

This pattern is related N.N.

Appendix C.7: Context Pattern #7, garment

… this pattern looks at contexts that are relevant for garment worn by the user. In particular the activities of the user are at the centre of the interest to this pattern.

The **entity** is a garment that can be worn be the user. Typically the artefacts do not incorporate any electronics to provide their functionality. Beyond the basic functions provided by garment they are also considered as a fashion statement.

Scenario: A user wears shoes, socks, underwear, trousers, a t-shirt, and a jacket. They are walking in the street. They walk through a door into a house. There they welcome a friend by giving them a hug. They put their jacket on the coat hanger and sit down.

Typical **examples** of such devices are shoes, socks, dresses, trousers, and shirts.

Contexts of interest include: garment is worn, garment is in the locker, garment is on the coat hanger, action of the person wearing then garment (dependent on the garment), arrangement of garments on the body, interaction with garments, and environmental conditions around.

The contexts mentioned above are **of interest to** devices in the body network of the user an also to some extent to the environment.

Major constraints and **forces** on the implementation include design, power consumption, robustness (e.g. sustain washing in a machine), weight, size, and unobtrusiveness.

The prime **sensing technologies** that are appropriate include: accelerometers (position and acceleration), bio-sensing, touch on the garment, temperature at different positions and light sensors.

The basic context, such as worn or in the locker, can be easily discriminated without learning. However complex contexts concerning the actions of the wearer are much more individual and therefore are difficult to be hard wired into the device. Here **perception methods** that offer flexible learning, such as self-organising maps, are a useful option.

The **device architecture** is highly depended on the design of the garment, which is one of the most important requirements for this pattern. For many sensing tasks it is inevitable to integrate the sensors with the fabrics of the garment, or at least distribute them over the garment. As power is a further concern processing power should be kept minimal and therefore low power MCU are appropriate. If possible within the anticipated design power harvesting can provide energy, [Starner,96], [Kymissis,98].

When the contexts are provided for other components in the body network a number of options for **communication** are available, such as wired bus, short range wireless, wired with self-arranging connectors, and using the skin as communication media. For communication to the environment a wireless connection is required; a central component- e.g. a wearable computer can act as communication hub also providing access control. Off-line communication is a further option for garments. If only a log of the activity is required the garment can store this and only in a deliberate act, e.g. by connecting it to a docking station the data is communicated.

Implementation examples are a context-aware shoe [Paradiso,00], context tie [Schmidt,99b], and a context-aware wearable computer using the distributed TEA approach [Laerhoven,00], [Laerhoven,02].

This pattern is related N.N.

Appendix C.8: Context Pattern #8, location awareness for mobile computing devices

… this pattern looks at location as a special case of context that is relevant for mobile computing devices. This pattern concentrates on where a device is used.

The **entity** is a mobile computing system that incorporates a computer and also a graphical or text-based user interface. The prime functionality is to ease and filter access to information and commands based on the whereabouts of the device and respectively of the user.

Scenario: A user carrying a PDA walks as tourist in a city that has certain spots which are particular interesting. When approaching the castle they get information displayed that is related to the place. Walking further outside a cinema they get information about the films that are on tonight and also an interface to order tickets.

Typical **examples** of such devices are tablet pc, mobile phone, PDAs, wearable computers, and augmented reality systems.

Contexts of interest include: the position of the device and the direction the device is facing.

The contexts mentioned above are primarily **of interest to** the device itself and more to the applications running on the device. The applications may communicate the contexts further.

Major constraints and **forces** on the implementation include precision and reliability, power consumption, robustness, weight, and size.

The prime **sensing technologies** are sensors that provide location (e.g. outdoors GPS, dGPS, and GSM/MPS; indoors RF and IR beacon systems). Furthermore a compass can provide orientation. Also accelerometers or gyroscopes may be appropriate to compensate for the movement of the device.

As the sensors above already provide position (e.g. in NMEA183-format) and orientation (e.g. the angle in º) in a symbolic representation, no low level perception is required. **Perception methods** on a higher level include usually rules that map

symbolic position and orientation information onto concepts that are used in the information domain, such as places.

The **device architecture** for this pattern is most often a computing device that is equipped with a component that provides the sensor information. This functionality can be realised in an add-on component or when communication infrastructure is used (e.g. GSM) this is already built in. In the case of a RF-beacon or IR-beacon based systems also built-in sensors (e.g. IrDA-port, Bluetooth transceiver) can be used.

When external modules are plugged into the device **communication** will be realised by wire, e.g. a PC-card with GPS and a GPS/compass modules attached via serial line. When internal sensors are used communication is realised with software.

Implementation examples are a context-aware tour guides and augmented reality experiences [Höllerer,99].

This pattern is related N.N.

Appendix C.9: Context Pattern #9, context aware recoding devices with communication

… this pattern looks at artefacts that record contexts over time as a specific type of context aware devices. This pattern concentrates on the recoding and tracking of contexts.

The **entity** is a mobile context recoding system that incorporates a sensing infrastructure, processing, storage, and communication. The prime functionality is to record contexts that appear over time and make them available at synchronisation points for further processing.

Scenario: A box of hard drives is prepared for shipping from Taiwan to the UK. The manufacturer puts a context recording device into the box that measures the temperature, humidity, magnetic field, and acceleration of the box constantly. When pre-specified thresholds are violated the sensor readings are saved together with a timestamp. The receiver of the goods reads out the contexts.

Typical **examples** of such devices are temperature, humidity, location, acceleration, and shock logging devices used in logistics.

Contexts of interest include: the violation of thresholds to the specific sensors related to the time.

The contexts mentioned above are primarily **of interest to** backend systems processing the transfer of goods. Logically the recorder is attached to the objects nearby performing extrinsic sensing.

Major constraints and **forces** on the implementation are price, robustness, reliability, power consumption, weight, and size.

The prime **sensing technologies** are sensors that provide temperature, humidity, location, acceleration, and magnetic field information.

Perception methods are in these cases simple and usually based on rules that observe whether or not specified conditions have been violated.

The **device architecture** for this pattern is a sensing module, a processing component, storage, and a communication module tied together in one device.

The **communication** is carried out at designated points and when defined events occur. When the device is designed to be synchronised at specific points only the communication can be based on wired connections, IR or short-range RF. When devices should have the ability to communicate a critical context without any delay a long-range communication technology, such as GSM, has to be included.

Implementation examples are intelligent tracking systems, as we explored in the project aware goods [Thede,01], [Patent,02].

This pattern is related N.N.

Appendix D: Building Blocks and Libraries

The following sections show selectively building blocks and libraries that are essential for the rapid prototyping platform introduced in chapter 5.

Appendix D.1: Hardware Building Blocks HWcore

Three different core modules for PIC microcontrollers [Microchip,02] are depicted in Figure 34.

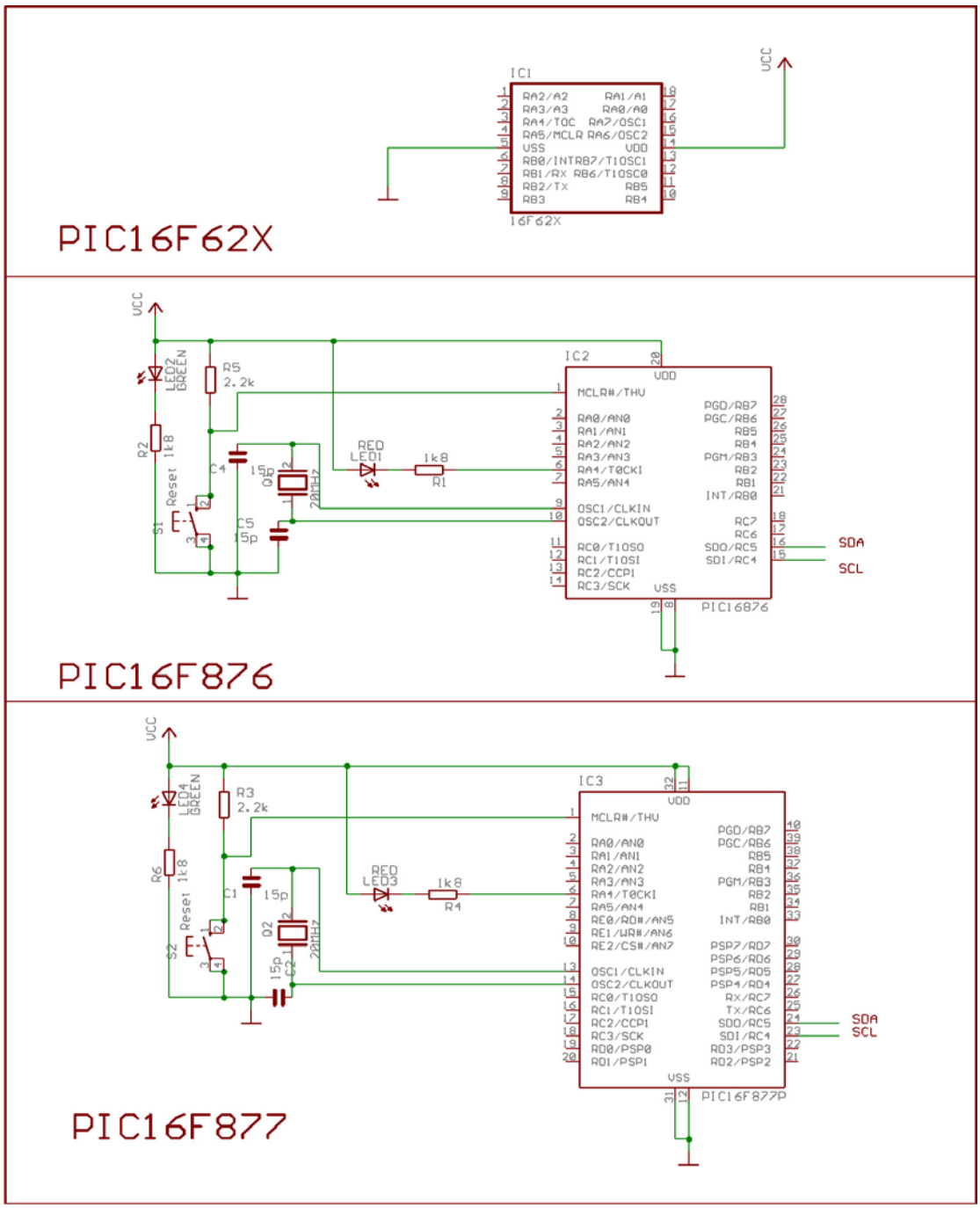


Figure 34: Selection of core hardware building blocks.

Appendix D.2: Sensor Building Blocks HWsensor

In this section the hardware building blocks of some of the most commonly used sensors are presented, see Figure 35.

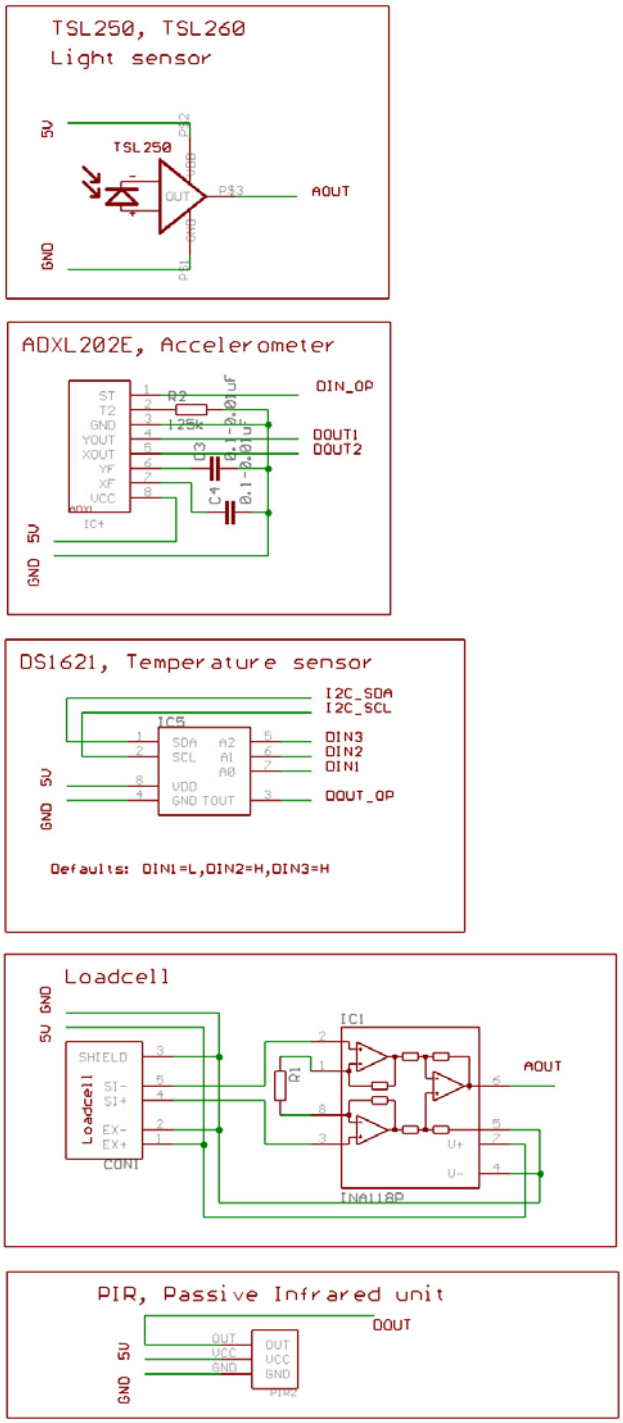


Figure 35: Sensor Building Blocks.

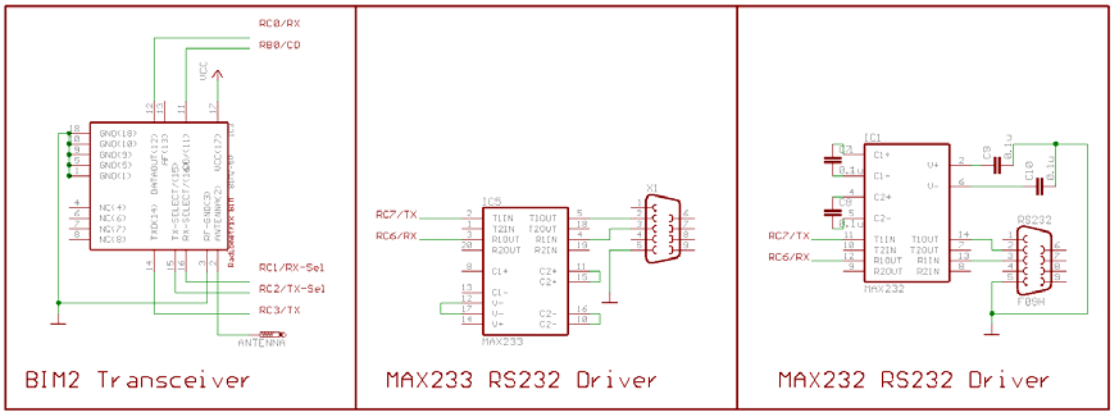


Figure 36: Communication Building blocks.

Appendix D.3: Communication Building Blocks HWcomm

The following building blocks show a wireless communication module and two options for connecting the MCU to a serial line, see Figure 36.

Appendix D.4: Core Libraries SWcore

The following functions are provided in the core library:

```
// switch the LED on and off
void led_on()
void led_off()

// print out a boot message and toggle LED
void boot_sign()

// initialize core functionality, e.g. watch dog timer
int init_core()

// access function, external ram
void write_ext_fram(long int address, byte data);
byte read_ext_fram(long int address);
```

Appendix D.5: Sensor Drivers SWsensor

To use the sensor drivers in the beginning of the program the connections pins must be defined:

```
// accelerometer output is connected to PIN_B5 and PIN_B6
#define PIN_ADXL_XOUT PIN_B6
#define PIN_ADXL_YOUT PIN_B5

// light output is connected to analog input 0
```

```
#define LIGHT_ANALOG 0

// PIR sensor is connected to PIN_B1
#define PIN_TOUCH PIN_B1

// Touch sensor is connected to PIN_B2
#define PIN_TOUCH PIN_B2
```

The following functions are a selection of primitives to access sensors.

```
// access full scale acceleration values
// PIN_ADXL_XOUT, PIN_ADXL_YOUT must be defined
unsigned long get_accelerationX();
unsigned long get_accelerationY();

// byte size acceleration
unsigned int get_accX();
unsigned int get_accY();

// read the analog light value, LIGHT_ANALOG must be defined
long light();

// read the passive infrared sensor, PIN_PIR must be defined
short int PIR1();

// read the touch sensor, PIN_TOUCH must be defined
short int touch();

// reads the current temperature, takes about 1 second
byte temp();
```

Appendix D.6: Communication Drivers SWcomm

To use the communication functions the pins to which the communication module is connected must be defined at the beginning of the program. The a set of high level functions for printing and receiving over RF is provided for the application programmer. These functions are based on a set of low level functions. The following examples shows this exemplarily for the Radiometrix BIM2 transceiver.

```
// connection pins
#define RF_TX_PIN PIN_C3
#define RF_RX_PIN PIN_C0
#define RF_CD_PIN PIN_B0
#define RF_TX_ENABLE_PIN PIN_C2
#define RF_RX_ENABLE_PIN PIN_C1
#define RF_SPEED 19200

// high level functions
```

```
// to be used by the application programmer
// clear buffer, use before printf
void reset_rf_buffer();
// the following function is used as first argument in printf
void to_rf_buffer(char c);
// send the buffered output in an RF package
void RF_printf();
// switch BIM2-module in power down mode
void rfPowerDown();
// switch BIM2-module in transmission mode
void rfTxOn();
// switch BIM2-module in receive mode
void rfRxOn();
//print the version over RF
void RF_version();

// support functions
// write a long to RF
void RF_put_long(unsigned long ldata);
//write/get a char to RF
void RF_putc(char data);
char RF_getc();
// check for a char on RF
int RF_kbhit();
// calculate a crc16 offline
unsigned long crc16(char * msg, int len)
// switch BIM2-module in self test mode
void rfSelfTest();
// transmit a number of characters via RF
int rfTransmit(char * msg, int len);
// receive a packet via RF
int rfReceive(char *buf, int maxLen);
// receive a packet via RF
int rfReceiveOnCD(char *buf, int maxLen, long timeOut)

// important global variable:
// rf_buffer
```

Appendix E: Schematics

In this section the full schematics off the main parts of the rapid prototyping platform are presented. These schematics are electronically available at http://www.comp.lancs.ac.uk/~albrecht/smart-its/platform.

Appendix E.1: Core Board Schematic.

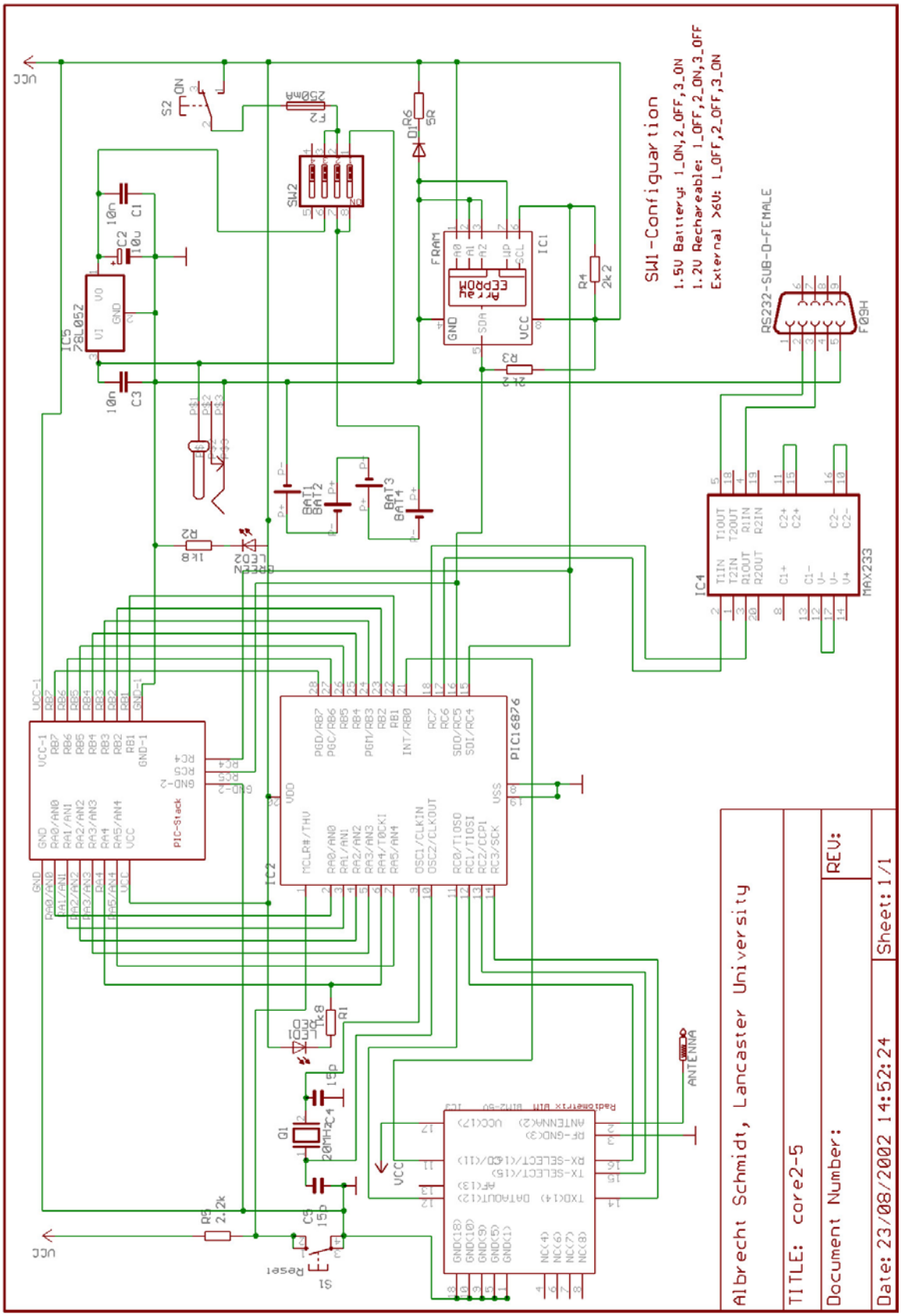


Figure 37: Schematic of the Smart-Its core board.

Appendix E.2: Mini Core Board Schematic.

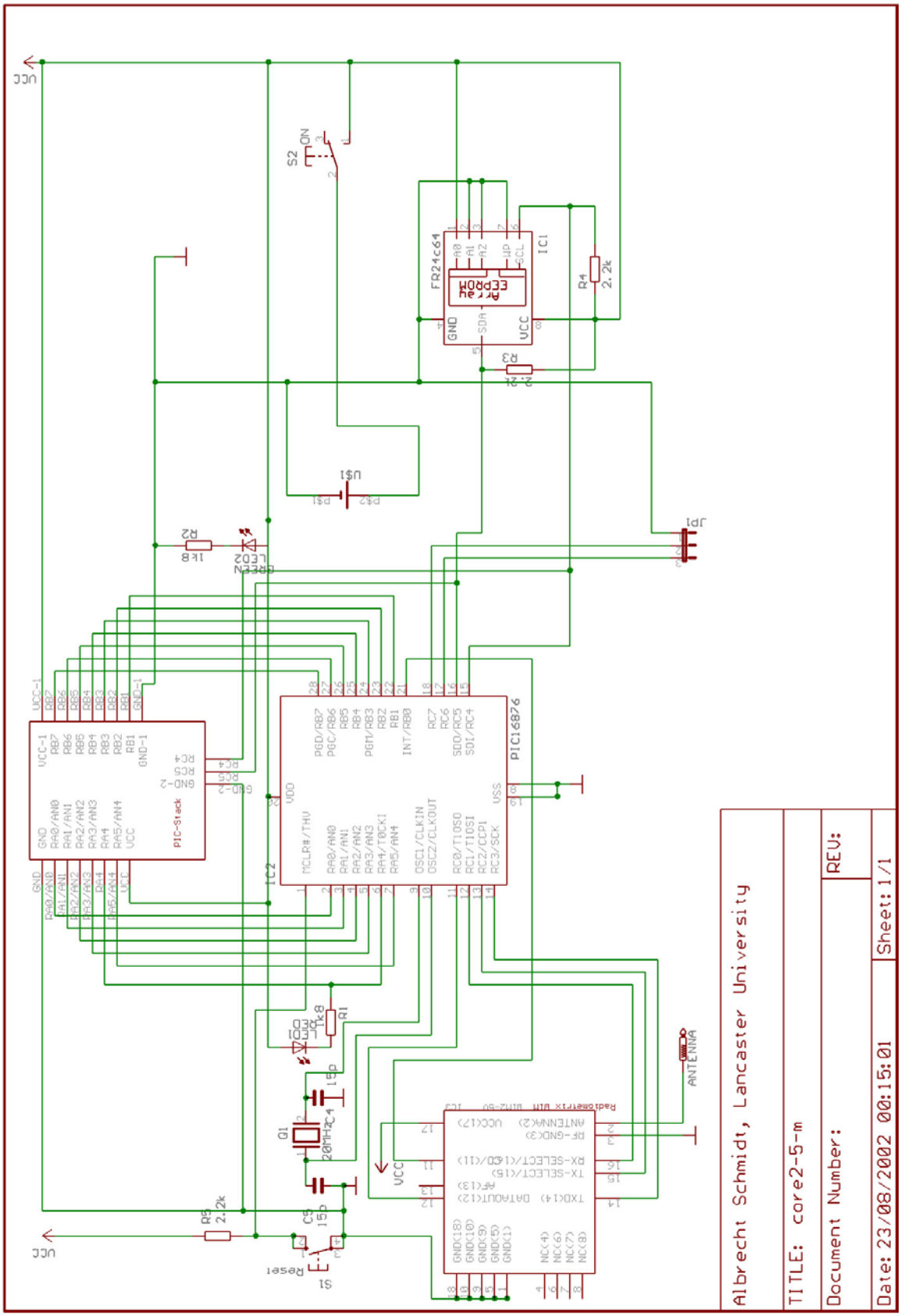


Figure 38: Schematic of the Mini-Smart-It core board.

Appendix E.3: General Sensor Board Schematic.

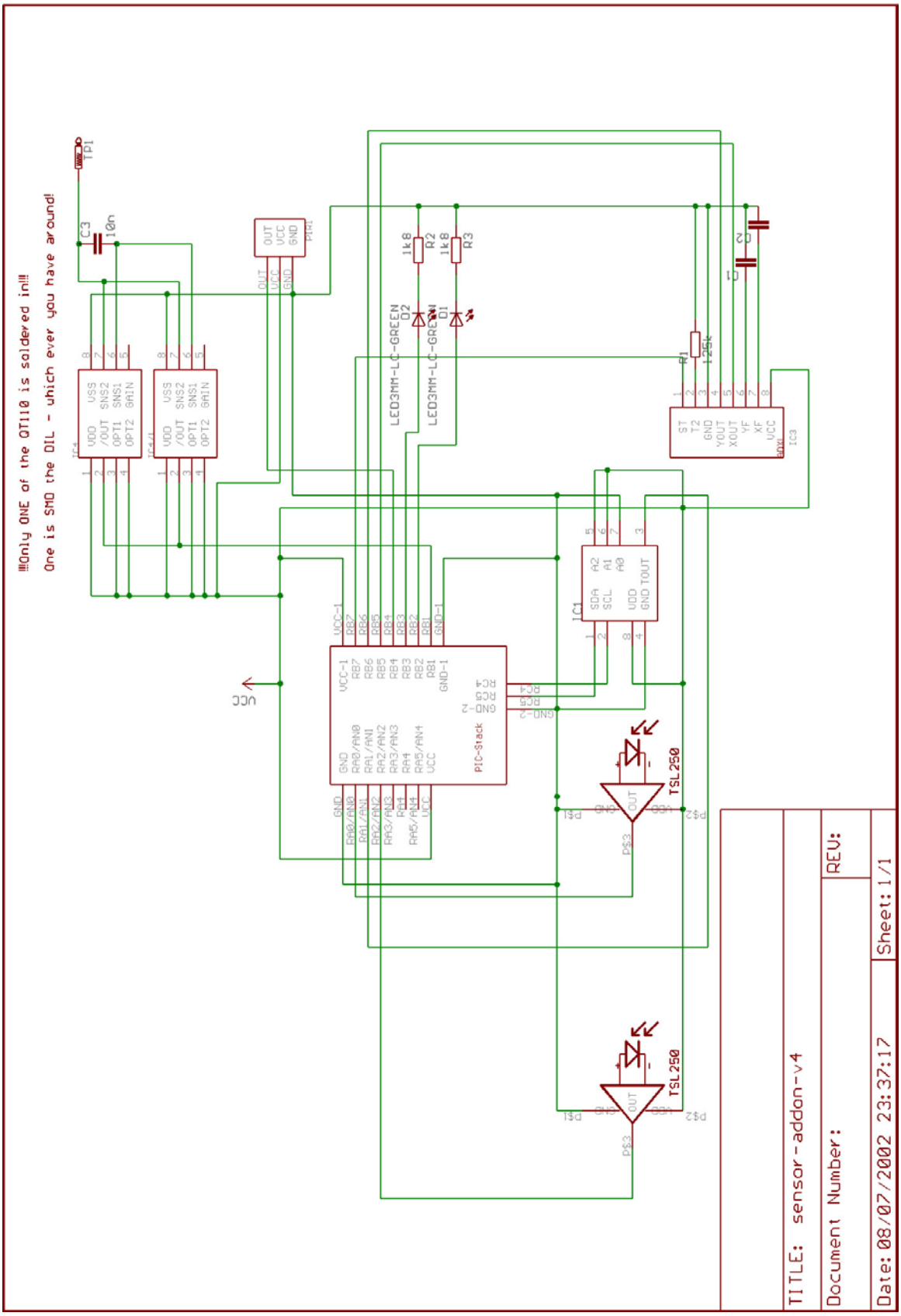


Figure 39: Schematic of the Sensor Add-On-Board.

Appendix E.4: Load Sensing Add-On Schematic.

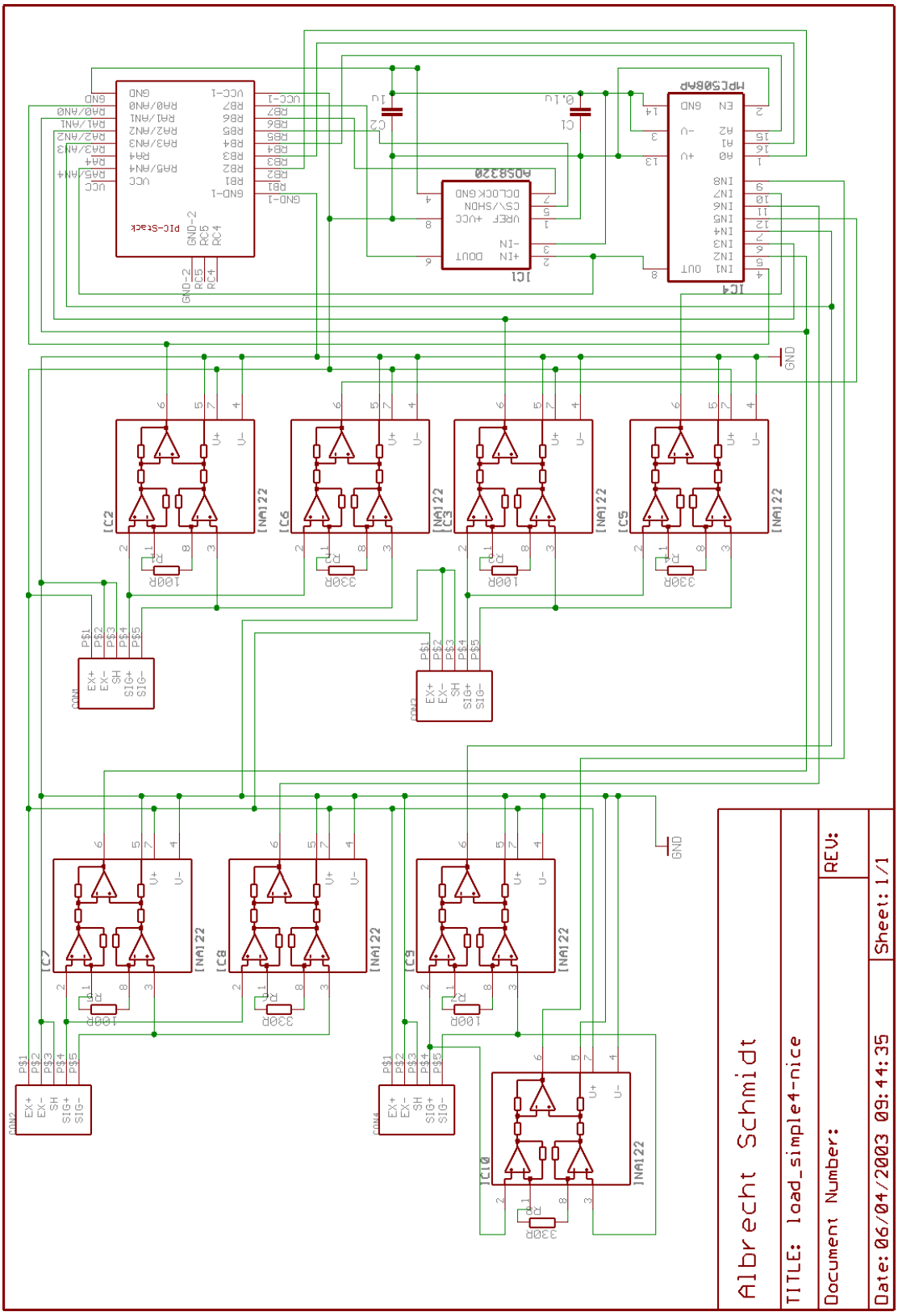


Figure 40: Schematic of the load sensing Add-On board.