Diss. ETH No. 16653

User-Centric Dependability Concepts for Ubiquitous Computing

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH (ETH ZURICH)

for the degree of

Doctor of Technical Sciences

presented by

Jürgen Bohn

Diplom-Informatiker, University of Karlsruhe (TH), Germany

born October 16, 1973

citizen of Germany

accepted on the recommendation of

Prof. Dr. Friedemann Mattern, examiner

Dr. Albrecht Schmidt, co-examiner

2006

Abstract

Ubiquitous computing has the goal of enhancing computer use by making many computers available throughout the physical environment, and thus providing the technical and conceptual means for enabling anytime, anywhere, anyhow computing. The technical realization of this vision has become feasible owing to recent advances in miniaturization and embedded computing technologies, which enable the integration of diverse computing capabilities into manifold everyday objects and devices.

The multitude and diversity of computerized entities that are part of ubiquitous computing systems imply a number of technical challenges. On one side, the differences in form factors, functionality, and technical capabilities result in a great level of diversity of hardware and software, and a distinct heterogeneity in capabilities. Further, ubiquitous computing systems in general are often highly decentralized, featuring an exceptionally high volatility of cooperative relationships and topologies due to mobility, spontaneous interaction, and wireless ad hoc communication on behalf of system components. On top of that, a particularly significant characteristic of ubiquitous computing is its unprecedented degree of *user-centricity*. Ubiquitous computing systems are no longer distinct entities separable from the human user, but instead explicitly designed to embrace and support people in everyday life settings and situations.

In the face of the technical issues and the strong user-centricity, coping with the *dependability* of ubiquitous computing systems and infrastructures constitutes a crucial challenge, especially since these systems are set to pervade and reshape people's everyday life environment and the way they complete their daily chores.

The goal of this dissertation is to provide a systematic study of critical dependability challenges in the context of ubiquitous computing, with particular consideration of the needs of the end-user, which to this point in time has not been investigated deeply in the research community. In doing so, we focus on two fundamental areas of ubiquitous computing: *human-computer interaction* and *context-aware computing*.

Dependability in the area of human-computer interaction so far mainly focused on the design and implementation of user interfaces. However, with the emergence of highly interactive ubiquitous computing environments, the user's dependence on interface devices has been increasing significantly even in everyday life situations. This leads to a new, user-centric type of dependability threat: given that a user requires certain user interface devices to interact with or to control a surrounding ubiquitous computing environment on a daily basis, it has to be investigated how he or she can cope with the unanticipated unavailability of these devices. It is therefore important to ensure the *physical availability* of user interface devices in order to preserve the *accessibility* of personal data and device functionality required as part of the human-computer dialogue.

A further fundamental challenge is dependable context- and location-aware com-

puting, as the manifold mobile or portable devices and smart objects found in ubiquitous computing environments are particularly subject to frequent and spontaneous changes in their surrounding context. Mobile devices not only have to cope with dynamic changes in locally available resources and in the physical properties of the user's environment, but they often also have to take into account unanticipated alterations of user-specific parameters such as the user's location, activity, personal preferences, and intentions. By enabling mobile devices to retrieve context autonomously in the respective location where it matters and using it as implicit input, context-aware computing provides means for more user-friendly, unobtrusive computing services. However, from the viewpoint of dependability, a crucial challenge is to provide mobile devices with reliable means of context awareness, and – in addition – to enable applications to use redundant context for the implementation of fault-tolerance mechanisms. Here an important task is to enable mobile devices and applications to tolerate and adapt at runtime to disturbances that are liable to occur in highly dynamic ubiquitous computing environments, such as the temporary or permanent unavailability of resources, the absence of network connectivity, or the unavailability of remote infrastructure-based services.

The main contributions of this dissertation are threefold. Firstly, we present concepts that increase the accessibility of personalized device functionality and improve the availability of personal user data. For that, we developed and prototypically implemented a system that exploits the diversity of user interface devices by means of a redundant *input/output diversification*. We further motivate the concept of *instant personalization and temporary ownership* as a method of freeing the user from the dependence on individual personalized devices while preserving the advantages of customized device functionality and access to personal user data.

Secondly, we describe concepts for improving the dependability of user-centric systems in ubiquitous computing environments by exploiting redundancy and diversity of resources, with a focus on location awareness. For that we have investigated and developed two concepts that enable fault-tolerant computing based on localized cooperation and resource sharing: (1) fault-tolerant services based on *cooperating smart everyday objects*, and (2) *super-distribution of smart entities*. The first approach enables mobile devices to exploit the multitude and diversity of volatile resources, which are found in the local computing environment of these devices, for the realization of fault-tolerant services. The idea of the second approach is to distribute computerized entities in a highly redundant way over object surfaces and physical environments, and to use the resulting physical infrastructure as a substrate for the realization of self-contained, fault-tolerant location-aware services and applications. As a further elaboration of the concept of redundancy, we present two systems that exploit diversity of sensor technologies and sensing capabilities by applying *sensor data fusion* techniques: (1) a lightweight and extensible system for the self-positioning of mobile devices based on an open sensor-fusion architecture, and (2) a solar-cell based positioning system that uses locally available context knowledge for real-time self-calibration and service optimization to improve the quality and robustness of the positioning procedure.

Last but not least, we identify and discuss dependability challenges of human-centered ubiquitous computing systems from a broader *social and ethical perspective*, including general aspects of reliability, control, social compatibility, and user acceptance.

Zusammenfassung

Die Nutzung von Computern ist im Begriff, sich aufgrund der zunehmenden Durchdringung unserer alltäglichen Lebenswelt mit einer ständig wachsenden Anzahl von computerisierten Gegenständen und Geräten grundlegend zu verändern. Der Entwicklung zugrunde liegt die Vision des *Ubiquitous Computing*: eine unaufdringliche Computernutzung an jedem Ort, zu jeder Zeit und auf verschiedenste Art und Weise zu ermöglichen. Technisch verwirklichbar wurde diese Vision in jüngster Zeit durch die Fortschritte in der Miniaturisierung und in der Entwicklung von eingebetteten Computersystemen, welche es erlauben, kleinste Computer in nahezu beliebigen Alltagsgegenständen und -geräten zu integrieren.

Die aufkommende Vielzahl und Vielfalt an computerisierten Objekten in alltäglichen Umgebungen stellt eine große technische Herausforderung für Ubiquitous-Computing-Systeme dar. Die zum Teil erheblichen Unterschiede in Bezug auf Formfaktor, Funktionalität und technischer Ausstattung führen zu einer gesteigerten Software- und Hardware-Diversität sowie zu einer ausgeprägten Verschiedenartigkeit (Heterogenität) der Fähigkeiten. Darüber hinaus sind Ubiquitous-Computing-Systeme in der Regel sehr dezentralisiert und durch eine hohe Unbeständigkeit (Volatilität) der kooperativen Wechselbeziehungen und Topologien gekennzeichnet, was in großem Maße durch die Mobilität, spontane Interaktion und drahtlose Ad-hoc-Kommunikation von Systemkomponenten bedingt ist. Hinzu kommt, dass Ubiquitous Computing in einem bisher nicht da gewesenen Ausmaß *benutzer-* und *humanzentriert* ist. Ubiquitous-Computing-Systeme sind keine klar vom menschlichen Benutzer trennbare Gebilde mehr. Vielmehr ist ihre ausdrückliche Zielsetzung, Menschen in alltäglichen Lebensumständen in unaufdringlicher Weise zu unterstützen.

Im Hinblick auf die technischen Schwierigkeiten und die ausgeprägte Benutzerzentriertheit von Ubiquitous Computing ist die Verlässlichkeit von Ubiquitous-Computing-Systemen und Infrastrukturen ein zentrales Anliegen, das letztendlich ein entscheidendes Kriterium für die Verbreitung, Benutzbarkeit und Akzeptanz von allgegenwärtigen Computersystemen bildet.

Ziel dieser Dissertation ist es, kritische Herausforderungen hinsichtlich der Verlässlichkeit von Ubiquitous Computing unter besonderer Berücksichtigung der Endbenutzer systematisch zu analysieren. Diese Problemstellung wurde in der Ubiquitous-Computing-Forschung bisher noch nicht eingehender untersucht. Dabei wird der Schwerpunkt auf die Interaktion von Mensch und Maschine sowie auf kontextabhängige Computersysteme gelegt.

Die Betrachtung von Verlässlichkeitsaspekten im Bereich der Mensch-Maschine-Interaktion hat sich bislang vorwiegend auf den Entwurf und die Entwicklung von Benutzerschnittstellen beschränkt. Mit dem Aufkommen hochinteraktiver Ubiquitous-Computing-Umgebungen nimmt jedoch die Abhängigkeit von Benutzergeräten in Alltagssituationen signifikant zu. Dies führt zu einer neuartigen Gefährdung der Verlässlichkeit aus Sicht des Anwenders: Damit ein Benutzer ein

ihn ständig umgebendes Ubiquitous-Computing-System kontrollieren bzw. mit ihm interagieren kann, benötigt er gewisse Benutzerschnittstellengeräte. Hier gilt es Konzepte zu entwickeln, die dem Anwender im Falle der Nichtverfügbarkeit von bevorzugten Benutzergeräten weiterhin den Zugriff auf persönliche Daten ermöglichen und die erforderliche Funktionalität sicherstellen. Einen wichtigen Aspekt bildet dabei die Identifikation und Nutzbarmachung von vorhandenen Redundanzen in Bezug auf Benutzerschnittstellen und -geräte.

Eine weitere grundsätzliche Herausforderung stellt die Verlässlichkeit von kontextabhängigen und ortsbewussten Computersystemen dar. Mobile und tragbare Geräte sind in Ubiquitous-Computing-Umgebungen häufig von spontanen Veränderungen in ihrem jeweiligen unmittelbaren Umfeld (Kontext) betroffen: sie müssen sich nicht nur auf häufige Veränderungen der lokal verfügbarer Ressourcen in der Benutzerumgebung einstellen, sondern oft auch unvorhergesehene Wechsel von benutzerspezifischen Kenngrößen berücksichtigen, wie z.B. den Ort des Benutzers, seine Aktivitäten, seine persönlichen Präferenzen und Absichten. Indem mobile Geräte befähigt werden, ihren Kontext selbstständig und autonom direkt am eigenen Ort zu erfassen, ermöglichen kontextabhängige Computersysteme die Bereitstellung von benutzerfreundlicheren, unaufdringlicheren Dienstleistungen.

Der Einsatz von Ubiquitous-Computing-Systemen erfordert daher eine zuverlässige Ausstattung mobiler Geräte mit Kontextinformationen. Zentraler Aspekt hierbei ist es, mobile Geräte und Anwendungen zu befähigen, sich an Störungen zur Laufzeit – wie sie in hoch dynamischen Ubiquitous-Computing-Umgebungen häufiger etwa in Form vorübergehender oder dauerhafter Nichtverfügbarkeit von Ressourcen, fehlender Konnektivität oder Nichterreichbarkeit entfernter infrastrukturbasierter Dienste anzutreffen sind – anzupassen, diese zu tolerieren, und redundante Kontextinformationen für Fehlertoleranzmaßnahmen einzusetzen.

Diese Dissertation setzt sich in drei Hauptbeiträgen mit der Untersuchung von Verlässlichkeitsaspekten in Bezug auf Ubiquitous Computing auseinander:

Im ersten Hauptteil werden Konzepte präsentiert, die die Verfügbarkeit von interaktiven Benutzerschnittstellen erhöhen, um insbesondere auch die Verfügbarkeit von personalisierter Gerätefunktionalität und persönlichen Benutzerdaten zu verbessern. Hierzu wurde ein System entwickelt und prototypisch implementiert, das auf die Verschiedenartigkeit von Benutzergeräten für eine redundante *Ein-/Ausgabediversifizierung* zurückgreift. Der vorgestellte Ansatz der *augenblicklichen Personalisierung und vorübergehenden Inbesitznahme* befreit den Benutzer aus der Abhängigkeit von individuellen persönlichen Geräten, bewahrt dabei jedoch gleichermaßen die Vorteile von benutzerangepasster Gerätefunktionalität und der Verfügbarkeit von persönlichen Daten.

Im zweiten Hauptteil werden diverse Ansätze zur Steigerung der Verlässlichkeit von kontextabhängigen Computersystemen in Ubiquitous-Computing-Umgebungen durch die Ausnutzung lokal vorhandener Ressourcenvielfalt und -redundanz besprochen. Der Schwerpunkt liegt hier auf den im Ubiquitous Computing bedeutsamen ortsbewussten Diensten und Anwendungen. Es werden zwei Konzepte vorgestellt, die durch lokale Zusammenarbeit und Ressourcenteilung fehlertolerante Dienste ermöglichen: (1) Fehlertolerante Dienste basierend auf *kooperierenden intelligenten Alltagsgegenständen* und (2) *Hochverteilung von intelligenten Einheiten*. Der erste Ansatz ermöglicht mobilen Geräten, die Vielzahl und Vielfalt an dynamisch veränderlichen Ressourcen in der unmittelbaren lokalen Umgebung

für Fehlertoleranzzwecke auszunutzen. Die Grundidee des zweiten Ansatzes ist es zunächst, computerisierte Einheiten hochredundant über die Oberfläche von Objekten oder physischen Umgebungen zu verteilen, um anschließend die so erhaltene materielle Infrastruktur als ein Substrat für die Entwicklung von unabhängigen fehlertoleranten ortsbewussten Diensten und Anwendungen zu nutzen. Als eine weitere Ausprägung des Redundanzansatzes werden zwei Systeme präsentiert, welche eine vorhandene Vielfalt von Sensortechnologien und sensorischen Fähigkeiten mit Hilfe von Sensorfusionstechniken nutzbar machen: (1) eine leichtgewichtiges und erweiterbares System für die Selbstpositionierung von mobilen Geräten, basierend auf einer offenen Sensorfusionsarchitektur, und (2) ein Solarzellen-basiertes Positionierungssystem, das lokal verfügbares Kontextwissen in Echtzeit für eine Selbstkalibrierung und Dienstgüteoptimierung zur Verbesserung von Genauigkeit und Robustheit der Positionsbestimmung einsetzt.

Im dritten und letzten Hauptteil wird die Problematik der Verlässlichkeit von Ubiquitous-Computing-Systemen aus einer weiter gefassten, ethisch-sozialen Perspektive beleuchtet und bewertet. Angesichts seiner ausgeprägten Human- und Benutzerzentriertheit werden abschließend mögliche Auswirkungen und Gefahren des Ubiquitous Computing analysiert sowie eine Reihe von sozialen Herausforderungen und Fragestellungen zu wichtigen Themenkomplexen wie Zuverlässigkeit, Kontrolle, soziale Kompatibilität oder Benutzerakzeptanz identifiziert und diskutiert.

Acknowledgments

If you want to build a ship, don't drum up people together to collect wood and don't assign them tasks and work, but rather teach them to long for the endless immensity of the sea.

Antoine de Saint-Exupery

I am deeply indebted to my supervisor, Prof. Friedemann Mattern, for the opportunity to work in his research group at ETH Zurich. Prof. Mattern gave me all the freedom of creativity I required for designing my own research "ship", and the encouragement, confidence, and support needed for building it. I am extremely grateful for his invaluable support and his remarks on preliminary versions of this dissertation. Further, I want to express my gratitude to my co-adviser Albrecht Schmidt for his highly appreciated support.

During my time at ETH Zurich I had the opportunity to work with extremely intelligent and kind people in the Distributed Systems research group, which is led by Prof. Friedemann Mattern. I would like to warmly thank all of them: Robert Adelmann, Ruedi Arnold, Vlad Coroama, Svetlana Domnitcheva, Christian Flörkemeier, Christian Frank, Oliver Kasten, Matthias Lampe, Marc Langheinrich, Matthias Ringwald, Christof Roduner, Michael Rohs, Kay Römer, Frank Siegemund, Silvia Santini, Thomas Schoch, and Harald Vogt.

My family and friends have always been a constant source of understanding and never-ending moral support. They were the ones who already saw my ship sailing when all I had in my hands were planks and bolts.

But above all, I wish to thank God for all the grace and favor he has bestowed on me: for granting me the will and the strength to persevere and overcome my own limitations, and for my beloved wife Giselle, who has become my greatest source of purpose, joy, and peace in this life.

Contents

1. Introduction

Dependability is a classical key challenge of distributed computing [Lap85, Lap92a, Jal94]. The dependability of a system is threatened if faults in components of the system occur, which in return are the cause for errors that are liable to lead to subsequent failure of the system and the services it provides. As the components of a distributed system are often physically distributed and loosely coupled, the sources of potential faults increase. Such faults potentially lead to an undependable system on whose services reliance cannot or will not be placed any longer.

Ubiquitous computing is a further development of the distributed computing idea. Ubiquitous computing systems typically feature a much higher degree of distributedness, heterogeneity, and system dynamics than we find in conventional distributed computing systems, providing manifold opportunities for errors to occur. In addition, ubiquitous computing systems are often characterized by a *multitude* or *abundance* of computerized entities that cooperate to some degree, and by a great *diversity* of resources, technologies, capabilities, and form factors. These characteristics further aggravate the classic dependability problem. Ubiquitous computing further aspires to pervade the everyday life environment of the end-user to a so far unprecedented extent [Wei93b]. Ubiquitous computing systems thus have a strong impact on the users' lives in daily situations [BCL+04a], and the associated dependability challenges hence exhibit a strong *user-centric*, *human-centered* character.

In ubiquitous computing, the occurrence of faults generally cannot be prevented, not even through meticulous validation and verification procedures. This is because individual devices or remote services are liable to spontaneously cease functioning or responding for various reasons, such as because of hardware failures, interference of wireless communication, energy shortage, unavailability of background communication infrastructures, user mobility, or deliberate user intervention (e.g., other users may turn their services/devices off or cancel cooperative device relationships without prior notice).

However, in certain situations, it is possible to tolerate faults during operation to maintain some level of operability. Such fault tolerance always requires *redundancy* of some kind, as the functionality of components required for providing a certain service needs to be taken over by other components that have both the capability and the capacity for doing so. The major *dependability challenge of ubiquitous computing* systems therefore can be rephrased into identifying and harnessing new and existing sources of redundancy for enabling fault-tolerant operation. Bearing this in mind, the multitude and diversity of resources in ubiquitous computing environments, which are often considered a hindrance of dependable systems development, can be turned into an opportunity and exploited as a valuable source of redundancy. At the same time, it is important to acknowledge that ubiquitous computing technologies and systems center on the individual user in his everyday life environment. In the long run, the user may not even be in a position to "opt out"

of a life largely influenced by ubiquitous computing systems [BCL+04a]. Consequently, dependability concepts for ubiquitous computing have to be user-centered and socially compatible, taking into account the particular needs and demands of the average user.

In this dissertation, we present a number of concepts for increasing the dependability of services and applications with regard to two fundamental areas of ubiquitous computing: human-computer-interaction and context-aware services. We show how the traditional concepts of redundancy and diversity can be applied to achieve fault tolerance in ubiquitous computing environments to provide more dependable services to the user. In doing so, we consider a service or system to be dependable from the user's perspective if it allows the user to perform a given task even in the presence of certain disturbances and failures. Such disturbances include the unavailability of network connectivity and of services of the background computing infrastructure, and the failure or unavailability of mobile devices the user requires for interaction with an omnipresent ubiquitous computing environment.

1.1. Problem Statement

The goal of this dissertation is to provide a systematic study of critical dependability challenges in the context of highly-distributed ubiquitous computing systems that are characterized by a multitude of smart cooperating and interacting objects. To this point in time this topic has not been investigated deeply in the research community.

In our work, we focus on two fundamental areas of ubiquitous computing: context-aware computing and human-computer interaction.

Firstly, microprocessors, memory modules, communication chips, and sensors are embedded into an ever broader range of everyday life objects. These objects thus become "smart" in the sense that they are capable of executing certain object-specific services and applications, and that they are able to interact and cooperate with other smart objects via wireless ad hoc or infrastructure-based communication. The ubiquity of such computerized entities and their sensory capabilities form the basis of *context-aware computing*. The main goal is to enable computer systems to gain knowledge and understanding about the particular situation of a person or a device (such as the current geographic or symbolic location, the user's activity or intention, or physical characteristics of the surrounding environment), and to use this knowledge as a source for implicit input to improve the quality and adaptability of computing services provided to the user. Here an important dependability challenge is to ensure the reliability and availability of user-centric context-aware services and applications that rely on a ubiquitous computing infrastructure. These services have to operate in an environment consisting of highly-distributed computerized entities that feature a great diversity of resources, technologies, capabilities, and form factors. Context-aware computing systems may also rely on remote services and databases situated in a networked background infrastructure or in the Internet. Here a major challenge is to maintain a certain functionality for the user even in the case that individual smart objects in the vicinity of the user and the services they provide fail to respond, or if remote infrastructure services and databases are unavailable due to lack of connectivity or failure.

Secondly, it has been recognized that computing systems are about to partially

dissolve into the environment and become much more intimately associated with their users' activities [HBC+96]. In this context, the interaction between users and their surrounding ubiquitous computing environments has become an important challenge of research in the domain of *human-computer interaction*. However, while much work has been dedicated to the development of novel interaction paradigms and user interfaces, the dependability aspect of the interaction with smart computing environments and mobile services so far has received little attention. As users more and more depend on mobile devices to perform everyday tasks at home, at work, or in public places, the interaction with surrounding ubiquitous computing infrastructures also has to be performed in a dependable manner. Concretely, there has to be explicit support for the user to ensure the accessibility of mobile devices and ubiquitous computing services even in situations where the user's customary physical tools of interaction, such as personal mobile devices, cease to operate or are physically unavailable.

1.2. Contributions

The main contributions of this dissertations are threefold. Firstly, we present concepts that increase the accessibility of personalized device functionality and improve the availability of personal user data by freeing the user from the dependence on individual mobile devices. Secondly, we describe concepts for improving the dependability of context-aware services in ubiquitous computing environments by exploiting redundancy and diversity of resources, with a focus on location awareness. Thirdly, we identify and discuss dependability issues from a broader social perspective. To demonstrate the feasibility and effectiveness of our concepts, we have developed several exemplary prototypical implementations, which we discuss in the respective chapters. Altogether, we present the following contributions:

1. **Concepts for dependable, device-independent interaction** between users and their surrounding smart environments by harnessing the following properties of ubiquitous computing environments:

   a) **Diversity of devices and communication technologies:**
   We demonstrate how the accessibility of personal user data and online services can be increased by means of *input/output diversification*. We have applied the concept in the prototypical implementation of the ETHOC System (EveryThing Has Online Content), where users can access online content and functionality linked to physical objects both by means of different personal devices and computing platforms, and by means of a generic web interface.

   b) **Multitude of mobile user devices:**
   We developed the concept of *instant personalization and temporary ownership*, which frees the user from the dependence on individual devices and provides mechanisms for protecting the confidentiality of personal user data. Here the main idea is to realize the *interchangeability* of devices while preserving the advantages of customized functionality provided by personalized handheld devices. As a proof of concept, we de-

scribe a working prototypical implementation for the use with personal digital assistants.

2. **Concepts for dependable location-aware services and applications** that exploit the following properties of ubiquitous computing environments:

   a) **Multitude of dedicated local resources:**

   We developed the concept of *super-distribution of smart entities*, where smart computerized entities are distributed in a dense and redundant fashion over object surfaces. We use the resulting dedicated infrastructure as the basis for a fault-tolerant service middleware which enables the realization of fault-tolerant location-aware services. We also describe a concrete realization of the concept based on RFID technology, together with reference implementations of exemplary middleware services and applications, such as fault-tolerant local data sharing, tracing and tracking, self-positioning, and collaborative map-making.

   b) **Multitude and diversity of volatile local resources:**

   While the super-distribution of smart entities constitutes a *dedicated* infrastructure, we also investigate means to provide fault-tolerant services to mobile devices by making use of the *volatile* resources obtained from heterogeneous smart objects and devices typically found in ubiquitous computing environments. As a result, we developed an infrastructure that provides mobile devices with fault-tolerant data dissemination and communication services by means of localized cooperation and resource sharing based on cooperating smart everyday objects.

   c) **Diversity of sensor technologies and sensing capabilities:**

   By virtue of *data fusion* techniques, mobile user devices can capitalize on the richness of diverse context data extracted from ubiquitous computing environments in order to improve the adaptiveness and fault tolerance of context-aware applications and services. We developed two systems for the self-positioning of mobile user devices that combine redundant sources of location context information: Firstly, we developed a positioning system for resource-limited mobile devices based on an open data fusion architecture. The system enables a mobile device to merge the context information (i.e., position information) inferred from the environment by means of multiple diverse sensors. This allows the device to tolerate the transient or permanent failure of individual sensors, enabling it to maintain a certain quality of service as long as some sensory input is available (graceful degradation). Secondly, we built a system that features real-time positioning and automatic self-calibration by combining light intensity measurements obtained from off-the-shelf solar cells with location and topology information retrieved in situ from densely distributed RFID tags. Here the combination of redundant data fusion with an online processing of location-dependent context knowledge improves the accuracy, availability, coverage, and energy efficiency of the positioning service.

3. **Investigation of dependability challenges for ubiquitous computing from a social perspective**:

   We identify and analyze dependability challenges of human-centered ubiquitous computing systems from a social and ethical angle, including general aspects of reliability, control, social compatibility, and user acceptance. We further discuss opportunities and pitfalls of living in a world of smart environments and augmented objects.

1.3. Structure of this Dissertation

The remainder of this dissertation is structured as follows:

In Part I, we review some *fundamentals* of dependability research (Chapter 2) and of ubiquitous computing in general (Chapter 3). Then we describe the relationship between the two disciplines by discussing basic dependability methods in the context of ubiquitous computing (Chapter 4). Finally, in Chapter 5, we motivate the challenge of *user-centric dependability* in the context of ubiquitous computing systems, which is central to this dissertation.

In Part II, Chapter 6, we first discuss the challenge of dependable human-computer interaction, and we describe two concepts we developed to address the dependability issues we have identified earlier: *input/output diversification* and *instant personalization*. In Chapters 7 and 8, respectively, we discuss prototypical system implementations of these concepts.

In Part III, Chapter 9, we further motivate the challenge of dependability with regard to context- and location-aware computing. We present three concepts for fault-tolerant location-aware computing: *localized cooperation and resource sharing based on smart everyday objects*, *super-distribution of smart entities*, and *redundant multi-sensor data fusion*. In Chapter 10, we describe a system based on physically proximate smart objects, which employs *localized cooperation and resource sharing* in order to achieve fault-tolerant operation on mobile devices.

The concept of *super-distribution of smart entities* we describe in greater detail in Chapters 11–13. In Chapter 11, we first present our work on super-distributed RFID tag infrastructures, explaining how a physical infrastructure of densely distributed RFID tags can be used for delivering novel location-aware services. Then, as a generalization of this work, we present the concept of *super-distribution of smart entities* in Chapter 12, together with a general fault-tolerant middleware architecture we developed. This is followed by a description of a number of prototypical reference implementations of this middleware based on RFID technology in Chapter 13.

We conclude Part III in Chapters 14 and 15, respectively, by describing two systems that apply *redundant sensor fusion* and *context knowledge* for achieving fault-tolerant positioning and automatic self-calibration.

In Part IV, Chapter 16, we analyze the dependability challenge of ubiquitous computing from a *social* perspective. Finally, in Part V, Chapter 17, we summarize the contributions of this dissertation, and we draw some conclusions.

Part VI contains the appendix. It provides complementary information on the basic terminology and concepts of dependability in general (Appendix A), and an overview of the background, technologies, and major research challenges of ubiquitous computing and its related disciplines (Appendix B).

Part I.

Fundamentals

2. Dependability

In this chapter we introduce the basic terminology and concepts of computing systems dependability. The chapter is complemented by Appendix A, which provides a more comprehensive overview of fundamental properties and concepts in the field of dependability.

2.1. Definition and Terminology

2.1.1. Definition

From a general perspective, *computing systems dependability* is considered a unified discipline with the explicit purpose of "designing, implementing, and using computer systems where faults are natural, foreseeable and tolerable" [AL86]. The term *dependability* itself is conventionally defined as the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers [Lap85, Lap92a]. In this context, the service delivered by a system is its behavior as it is perceived by its users, while the user may be human or another physical system.

2.1.2. Attributes of Dependability

Laprie defines fundamental *attributes* which emphasize different, complementary *properties* of dependability [Lap92a], and which serve to express the properties that are expected of a dependable system. These attributes allow us to assess the system quality in relation to system impairments and the means opposing them. The most significant attributes of dependability are *reliability*, *availability*, *safety*, and *security* [Lap85, Lap92a]. Reliability deals with the property of *continuity of service*, availability with *readiness for usage*, safety with *avoidance of catastrophic consequences*, and security with *prevention of unauthorized access and/or handling of information*.

2.1.3. Faults, Errors, and Failures

The term fault is usually used to name a defect at the lowest level of abstraction, such as a memory cell that always returns the value 0, for example. The dependability of a system is threatened if *faults* in components of the system occur, being the cause for *errors* which in return are *liable* to lead to subsequent failure. A *failure* manifests itself through a system behavior that is not compliant with the specifications, the latter being an agreed description of the system's expected function and/or service [Jal94].

2.1.4. Dependability Methods

The development and maintenance of dependable computing systems calls for suitable *means*. According to Laprie, dependability can be achieved by the combined utilization of a set of methods that can be classified into the following four groups: (1) methods to prevent the occurrence or introduction of faults (*fault prevention*); (2) methods to provide a service complying with the service specification even in the presence of faults (*fault tolerance*); (3) methods to reduce the presence (in terms of number or seriousness) of faults (*fault removal*); (4) methods that estimate the present number, the future incidence, and the consequences of faults (*fault forecasting*).

Fault prevention and fault tolerance are used to *provide* a system with the ability to deliver a service complying with the specification, thus constituting "dependability procurement". In contrast, fault removal and fault forecasting are more concerned with the "validation" of the dependability of the system.

Fault prevention tries to eliminate as many sources for faults as possible before the system is put in regular use without the deployment of redundancy. Fault tolerance, in contrast, uses *protective redundancy* [Jal94] to automatically mask failures and to avert system failure in case some components fail.

2.2. Dependability in Distributed Computing

Dependability is a classical key challenge of distributed computing [Lap85, Lap92a, Jal94]. When modeling processes in the real world, one often finds that different sub tasks have to be carried out in different locations, looking at production chains in manufacturing plants, for instance. Hardware, software, and data may therefore be physically distributed across several locations and operate self-sufficiently on local tasks, thus forming a distributed system. For a distributed system to be capable of achieving a global task efficiently, the different distributed pieces of hardware have to be able to reliably communicate to be in a position to cooperate and coordinate the different sub tasks of the superordinate process.

As the components of a distributed system are often physically distributed and loosely coupled, the sources of potential faults increase. Such faults are liable to cause errors and failures, which potentially leads to an undependable system on whose services reliance cannot or will not be placed any longer.

Traditionally, distributed systems are viewed in two ways, either as defined by its physical components, or as defined from the point of view of processing or computation. In the first case, we speak of the physical model of the system, and in the latter case of the logical model [Jal94].

2.2.1. Physical Model

In the physical model, the distributed system is regarded as a *physical network* of many autonomous computing entities, which are also called nodes. The nodes are geographically at different locations and connected with each other by means of a communication network, through which the nodes communicate with each other by exchanging messages. Each node is equipped with a processor, which has some private volatile memory, a private clock that is used for coordinating the

internal execution of instructions, a network interface through which the node is connected to the communication network, and software that governs the sequence of instructions to be executed on the node [Jal94]. These components of a node are considered to be *atomic*. Fault tolerance mechanisms in distributed systems aim at masking the failure of some of these components to prevent the entire distributed system from failing [Jal94]. Often a distributed system is modeled as consisting of nodes and of a communication network as the basic components. In this case, the main component failures that have to be addressed by fault-tolerance mechanisms are the failure of a node and the failure of the communication network.

2.2.2. Logical Model

The *logical model* concentrates on the applications viewpoint of a distributed computing system: a *distributed application* consists of a finite number of concurrently executing processes that cooperate with each other to perform some task [Jal94]. A process is defined as the execution of a sequential program, which in return is a list of statements or instructions. Further, concurrent processes can either be executed on a single processor, or in parallel on different nodes in the system, which is the more interesting case for a distributed application. Concurrent processes are *competing* if they share resources but at the same time do not exchange information between themselves. This corresponds to a set of independent processes. *Cooperating processes* are characterized by an exchange of information which is either based on message passing or on using shared data objects. In the context of distributed systems as they are traditionally seen, only message passing is possible, as no shared data is allowed, and processes are usually considered to be cooperating [Jal94]. From the applications point of view, the underlying network is treated as a *fully connected network*, assuming the physical network is connected. This implies that a message can be sent from one node to any other node, which is supported by suitable communication protocols. Consequently, the network topology is not considered at this level. The logical connection between any two processes which interact by means of message passing is called a *channel*. A channel is assumed to have infinite buffer, to be error-free, and to deliver messages in the order that they have been sent (which is ensured by underlying communication protocols).

2.2.3. Interrelationship of the Models

With regard to the dependability of a distributed system, failures that occur in the physical system can cause the failure of components in the logical system. If a physical node fails, it may cause the failure of some processes, which can be considered as logical nodes. Similarly, the failure of communication lines in the physical network may cause the failure of logical channels. According to Jalote, the primary goal of fault tolerance is to preserve some properties in the logical model, which ultimately includes the provisioning of services according to the applying specifications, despite some failures in the physical model [Jal94].

2.3. Fault Tolerance Through Redundancy

In order to make a distributed system fault-tolerant, *redundancy* of some kind has to be employed [Gär99]. A non-redundant system may be able to detect the failure of a vital component, but will not be able to recover from that failure due to a lack of spare or substitute components that could take over the missing functionality. However, if redundancy is added to the structure of the system (*structural redundancy*), the dependability of a system evolves from mere "fault detection" capabilities to "fault tolerance". A more formal definition of structural redundancy was provided by Geffroy and Motet [GM02], stating that a system features structural redundancy "if its structure possesses certain elements which are not necessary to the obtaining of a behavior conform to the specifications, assuming that all the structure elements have a correct functioning". According to Geffroy and Motet, if structural redundancy is to be implemented, the redundant resources can be expressed in terms of *hardware* (electronic components, integrated circuits, logical gates, etc.), *software* (statements, functions, procedures, data, or objects), and *time* (execution time of the algorithm and/or circuit).

2.3.1. Replication

A common method of achieving structural redundancy in a distributed system is to duplicate a physical or logical component to obtain multiple redundant copies (replication). Then, by simultaneously executing the redundant components, the individual results are compared and a correct result is chosen (for instance, see N modular redundancy (NMR) in the Appendix A.3). Alternatively, it is possible to only activate and use available backup-components if an error is detected in the currently active (or primary) component (cf. Appendix A.4.2). This procedure is also referred to as *forward recovery*. Rather than replicating a hardware or software component, a component that has reached an erroneous state can also be resumed from a recorded previous state that is known to be error free. This procedure makes use of *temporal* redundancy. A common example is to simply reset and restart a component, such as rebooting a computer after an error occurred.

2.3.2. Design Diversity

The repeated execution of a failed component or its substitution with identical replicas does not provide fault tolerance if the inherent *design* of the component is flawed. In this case, the execution of the flawed component and of all its replicas will fail, given that the particular input or boundary conditions expose the internal design fault and repeatedly lead to identical errors and the failure of the component or its provided service in the distributed system. While the root cause of failure of hardware components is some physical failure, software has no physical properties but is a totally conceptual entity. For that reason, software faults are always *design faults* [Jal94]. And while physical failures are usually caused by natural laws, software faults are the result of incorrect design or the presence of "bugs", both of which are caused by human errors.

To avoid identical errors caused by design faults, *design diversity* is a potentially effective method. Here the basic idea is that N independently designed software or hardware components are used rather than identical copies [AL86].

3. Ubiquitous Computing

In this chapter we present an overview of fundamental properties and characteristics of ubiquitous computing in general. This chapter is complemented by Appendix B, which provides a more comprehensive overview of the background, fundamental properties, and research challenges of ubiquitous computing.

3.1. Vision and Background

3.1.1. Vision

Mark Weiser was the first to describe the vision of *ubiquitous computing*, which "has as its goal the enhancing computer use by making many computers available throughout the physical environment", and making computers "effectively invisible to the user" [Wei93b]. Weiser observed that – unlike virtual reality – "ubiquitous computing endeavors to integrate information displays into the everyday physical world", that it aims at augmenting "the nuances of the real world", envisioning "a world of fully connected devices, with cheap wireless networks everywhere". And in contrast to conventional computing systems, ubiquitous computing has the explicit aspiration to transform the real world in every day life situations by providing the technical and conceptual means for enabling anytime, anywhere, anyhow computing.

3.1.2. Background

The forthcoming realization of the ubiquitous computing vision asserts itself in the continuing progress in the development of small and cheap computing and communication technologies [HMNS01]. This development largely benefits from the technological advances in the domain of *embedded computing* [Mat05]: as the power of microprocessors, storage capacities and communication bandwidth rapidly increase, it becomes technically feasible to build ever smaller, cheaper and more abundant computers. This development ultimately results in the creation of so called "smart things" which not only have access to the Internet and its plentiful resources, but which also are increasingly capable of autonomous cooperation and interaction with each other [Mat03].

Embedded Computing

Embedded computing systems are generally defined as special-purpose computer systems which are completely encapsulated by the devices they control. An embedded system usually has specific requirements and performs pre-defined tasks, unlike a general-purpose personal computer. Important examples where embedded systems are employed are sensor networks, fly-by-wire systems, engine control

(e.g., improved fuel efficiency and lower emissions in automobiles), medical implants and monitoring systems, avionics (e.g., navigation and collision avoidance), smart homes and work spaces, and space control. But there are also more mundane examples of how embedded systems pervade our everyday life environment, spanning the whole spectrum of electronic devices and products, such as electronic toys, multimedia entertainment systems, mobile (smart) phones, digital alarm clocks, and even toasters and coffee machines with built-in or embedded computer systems.

Distributed Computing

Obviously, ubiquitous computing constitutes a further development of the traditional distributed computing idea, promoting an ongoing process of decentralization [HMNS01]: the computer is "irresistible on its way to push all limits and is getting omnipresent", eventually becoming a "part of everyday life and an inevitable component when performing a variety of private and business related tasks". Beyond the era of personal and distributed computing, ubiquitous computing makes "information access and processing easily available for everyone from everywhere at any time", enabling users to "exchange and retrieve information they need quickly, efficiently, and effortlessly, regardless of their physical location".

However, compared to conventional distributed computing systems, ubiquitous computing systems typically feature a higher degree of distributedness, heterogeneity, and system dynamics. In contrast to the traditional distributed systems model where it is assumed that the underlying physical network is connected and thus also the logical nodes or applications are fully interconnected, physical and logical entities are much more loosely coupled in ubiquitous computing systems. Further, mobility, portability, and ad hoc wireless interconnectivity of smart devices effectuate an unprecedented degree of volatility of cooperative relationships and topologies.

Mobile Computing

Mobile computing can be considered the logical forerunner of ubiquitous computing. Since motion is an integral part of everyday life, ubiquitous computing technology must support mobility. If this is not the case, a user will be acutely aware of the technology by its absence when he moves [Sat01]. To satisfy the increasing desire for ubiquitous access to information, anywhere, anyplace, and anytime, adequate communication systems and software infrastructures are required in addition to mobile and portable devices [MS03].

3.1.3. Human-Centered Technology

A particular characteristic of ubiquitous computing is user-centricity: ubiquitous computing constitutes a *human-centered technology* [Mat03], which aims to provide ubiquitous access to information, communication, and computation by having users employ many different mobile, stationary and embedded computers over the course of the day. Consequently, ubiquitous computing focuses on mobile people and not just on mobile computers [SAW94].

3.2. User-Centric Ubiquitous Computing Challenges

In the area of ubiquitous computing, two fundamental user-centric and interaction-based research challenges have been established [AM00]: context-aware computing and human-computer interaction.

3.2.1. Context-Aware Computing

Context-aware computing aims at enabling computer systems to gain a certain knowledge and understanding about the particular situation of a person or a device, and to use this knowledge as a source for implicit input to improve the quality and adaptability of computing services provided to the user [Dey01]. Context-aware computing enables a software system to continuously adapt its behavior to a changing environment over which it has little or no control [RJH02]. Context awareness also facilitates automation, making the interaction with computing devices is simplified and more casual [Nel98].

Context awareness is a particular challenge for mobile devices and ubiquitous computing applications to enable adaptability in a physical and computational environment that is liable to change frequently [AM00]. Moreover, context-aware computing systems can also promote and mediate people's interactions with devices, computers, and other people, and it can help to navigate in unfamiliar places [SAW94]. Schilit et al. [SAW94] have classified context-aware applications into four major categories: proximate selection as an interface technique facilitating the interaction with nearby objects, automatic contextual reconfiguration of system components, contextual information and commands, and context-triggered actions.

Last but not least, context awareness is a *user-centric* view of computing [Nel98]. It enables computing systems to dynamically and non-intrusively adapt itself to a user's circumstances, reducing the focus on individual computing devices.

3.2.2. Location-Aware Computing

If context is narrowed down to any information that (1) can be used to characterize the particular location of an entity or that (2) is characteristic of a particular location, we talk of *location context*. Such location context can be used in two different ways: (1) *directly* for the localization and tracking of objects or people, for creating geographic or topological maps, and for the implementation of navigation systems in general, or (2) *indirectly* for providing location-based services which have the goal of delivering location-aware content to subscribers on the basis of the positioning capability of the wireless infrastructure [CCR+04].

For instance, Kang et al. developed an algorithm that provides mobile devices with a user-level notion of "place" by inferring semantic information about the user's current location [KWSB05], such as "my place of work", "the place we live", or "my favorite lunch spot", for example. This semantic location information can then be used for the development of location-aware systems, such as a location-aware cell phone that switches to a silent mode when its owner enters a place where

a ringer is inappropriate (e.g., a movie theater, a lecture hall, a place for personal reflection).

In general, location awareness is of particular interest to ubiquitous computing. Location information has been identified to remain "the single most important piece of context used in ubicomp applications" [ABO02]. As a consequence, numerous location aware systems have been conceived in ubiquitous computing [HB01], and location awareness still constitutes a major research challenge as of today [SLP05].

3.2.3. Human-Computer Interaction

The research field that investigates issues related to novel human-computer interfaces and interaction paradigms is generally called human-computer interaction (HCI). Ubiquitous computing technologies provide the technical foundations for advanced input and output devices as well as for ever more sophisticated small-scale embedded systems. In the process, physical objects play a central role as both physical representations and controls for digital information. This is mirrored in a "wave of new HCI research into ways to link the physical and digital worlds" [UI00], which aims at exploring the relationship between physical representation and digital information, and at highlighting new kinds of interaction that are not readily described by existing frameworks. These human-computer interfaces are typically referred to as *graspable interfaces* [FIB95] or *tangible user interfaces* [IU97]. An important challenge of tangible user interfaces is the seamless integration of representation and control, which differs markedly from the mainstream graphical user interface (GUI) approaches of modern human-computer interaction.

A commonly observed trend is that computing systems appear to partially dissolve into the environment and become much more intimately associated with their users' activities [HBC+96]. With the help of ubiquitous computing technologies, it becomes feasible to integrate computation and communication means into all kinds of objects and artifacts for which uses can be found. For this reason, the user finds the surrounding everyday life environment to be increasingly populated by smart embedded devices that may interact with each other or with the user. The resulting human interfaces to these embedded devices are often very different from those appropriate to conventional computers and workstations [HBC+96]. The increasing desire for novel, more *natural* user interfaces [AM00] becomes apparent in the broad spectrum of novel manipulative or haptic user interfaces and interaction paradigms that have been presented over the past years, such as [Rek97, HFG+98, KL99, KRA99, RAKO03], for instance. In the context of ubiquitous computing, the realization of anytime, anywhere accessibility of services by means of ubiquitous communication and user interfaces, and the integration of novel means of embedded computation in the process, constitute crucial challenges [HBC+96].

A prominent example for embedding computing technology into mundane everyday objects is the Mediacup [BGS01]. The idea here was to augment an everyday artifact, represented by an inanimate coffee cup, so that it makes information about itself available for computing within a local environment, assuming that such context is of primarily local value. The augmented cup can principally be used as a haptic user interface for explicit interaction (e.g., using the cup as a pointer device). By harnessing the information derived from the sensors monitoring the

state of the cup (artifact context) as a source of implicit input, the Mediacup can also be employed for the realization of context-aware applications, such as a smart door-plate application that indicates an ongoing meeting in a room and detects the participating persons depending on the presence and states of individual cups detected within the room, for instance.

3.3. A Characterization of Ubiquitous Computing Systems

The concept of ubiquitous computing, based on Weiser's visions and ideas [Wei91, Wei93b], does not provide precise instructions on how to build a concrete system. In other words, there is no one model for ubiquitous computing systems. Ubiquitous computing rather describes a system design and development paradigm (or "philosophy"). It strives to integrate a number of established disciplines (cf. Appendix B.2), such as embedded, mobile, peer-to-peer, and grid computing. It also spawned a number of more recent sub-disciplines, the most prominent of which probably are context- and location-aware computing (cf. Sections 3.2.1 and 3.2.2, respectively), human-computer interaction (cf. Sections 3.2.3), and sensor networks (cf. Appendix B.6).

Despite being this amalgam of disciplines and influences, ubiquitous computing systems can be characterized by a number of comprehensive properties, which were partly inherited from its related areas, or which emerged anew during the process of realizing Weiser's vision. Based on an analysis of literature in the field of ubiquitous computing, we identify and describe fundamental characteristics of ubiquitous computing environments and the applications that form or are part of those environments.

3.3.1. Decentralization

Ubiquitous computing systems are highly decentralized, distributing the responsibilities between manifold small, autonomous entities. Each entity takes over special tasks and functionalities, thus contributing to a heterogeneous computing landscape [HMNS01]. Decentralization usually requires additional efforts in data synchronization, to keep sources and destinations of information, which can be distributed across different devices and applications, consistent and up-to-date. Further, ubiquitous computing devices and applications are often embedded into a background service infrastructure, which provides management and system maintenance functionality (such as remote software deployment and update, for instance), or persistent, reliable data bases and mass storage capabilities. The back-end systems themselves must be highly scalable and flexible, potentially serving millions of ubiquitous computing entities, which can be delivered by means of established conventional architectures for distributed systems, such as client/server architectures or redundant server clusters employing load-balancing and fault-tolerance techniques, as described in detail by Tel [Tel00], for instance.

3.3.2. Pervasiveness

The advances in miniaturization and embedded computing technology paved the way for the practical realization of Weiser's vision of weaving computing technologies "into the fabric of everyday life until they are indistinguishable from it" [Wei91]. To phrase it in a more mundane way, it became possible to enhance everyday life objects and spaces with embedded computing technology and communication capabilities, thus making them smart [Mat03, Mat05]. By embedding computing technology in building infrastructure, we obtain interactive "smart spaces" [Sat01], which provide additional means of sensing and controlling the physical environment to smart objects and applications. So the characteristic of *pervasiveness* refers to the process of pervading and augmenting physical objects and spaces with (embedded) computing technology. The degree of pervasiveness of a ubiquitous computing environment is determined by the rate of penetration of ubiquitous computing technology into the (mobile and stationary) infrastructure.

3.3.3. Ubiquity

As Weiser foresaw "a world of fully connected devices, with cheap wireless networks everywhere" [Wei91], the availability of ever smaller and cheaper computing and communication technologies [HMNS01] gradually enables a large-scale, exhaustive deployment of objects and devices enhanced with ubiquitous computing technology. *Ubiquity* describes the characteristic of "anywhere" computing, referring to the property that ubiquitous computing technology does not just occur in individual places, but instead in abundant quantities "throughout the physical environment", with a particular stress on *everyday life situations*, as envisioned by Weiser [Wei93b]. So ubiquity is a result of the *multitude* and *abundance* of smart objects and devices that are found in the user's environment. What makes the proliferation of smart objects and devices possible at a large scale is the fact that prices for microelectronic functionality have been falling radically over the last years, and experts expect that this trend will continue for many years [Mat04]. This has the effect that computer processors and storage components are becoming ever more powerful, smaller, and cheaper at the same time, enabling the mass-production of a variety of novel low-cost computerized products for domestic use. A prominent example of ubiquity is the massive proliferation of mobile phones (and of mobile phone access points). Today mobile phones are sold in large quantities and have acquired the status of basic commodities, whereas only a few years ago, they still had the reputation of being expensive, clumsy status symbols with limited functionality [Mat04]. According to Gartner, in 2004 mobile phones were sold in quantities surpassing six hundred of millions of units worldwide [Gar05a], with sales expected to exceed 730 million units in 2005. Further, Informa expected the global wireless market to hit 2 billion subscribers by end of 2005 [inf05], after having reached 1.5 billion in mid 2004 [inf04], which corresponds to a global mobile penetration of about 30%. A similarly rapid growth can be observed at the market for personal digital assistants (PDAs). For instance, in the first quarter of 2005, PDA shipments increased 25 percent within one year to 3.4 million [Gar05b, KCMT05].

3.3.4. Diversification

In contrast to the all-purpose workstation of the past decades, where different applications were provided by means of different software, we find a variety of diversified devices in ubiquitous computing environments that suit a specific group of users for a specific purpose [HMNS01]. The *diversity of hardware and software* results in a high degree of *heterogeneity in capabilities* [HMNS01] (e.g., concerning communication interfaces and protocols, memory storage, energy supply, processing power, operating systems, and available operating system support and application software). It also contributes to an "uneven conditioning" [Sat01] of the environment, which is determined by a varying rate of penetration of ubiquitous computing technology into the infrastructure (i.e., a varying degree of pervasiveness), which is liable to effect significant differences in the "smartness" of different environments.

3.3.5. Portability

Portability can be defined as the ease with which a system or component can be transferred from one hardware or software environment to another [IoEEE90]. In ubiquitous computing settings, the portability of devices and objects plays an important role. On the one hand, this is a result of the shift from using self-contained and statically networked computers to the use of portable computers with wireless communication capabilities [FZ94]. On the other hand, portability is a fundamental concept in ubiquitous computing, as the latter aims at turning everyday artifacts into "computer embodiments [...] of many sizes and shapes, including tiny inexpensive ones that could bring computing to everyone" [Wei93a]. Nowadays the continuing advances in embedded computing and the thus enabled increasingly high pervasiveness of ubiquitous computing technology, together with the observed desire for specialized new gadgets on behalf of the consumers [HMNS01], spur the development of an ever greater variety of devices and augmented objects that are small and handy enough to be portable while retaining their usefulness and effectiveness for their particular purposes.

3.3.6. Mobility

Motion is an integral part of everyday life. Ubiquitous computing technology therefore must support *mobility*, or a user will be acutely aware of the technology by its absence when he moves [Sat01]. This is not only true for portable ubiquitous computing equipment carried along or worn on the body by the user, but also for ubiquitous computing devices in general that are part of a mobile carrier substance or of a vehicle, such as devices that are built into automobiles, airplanes, or trains.

3.3.7. Interconnectivity

Interconnectivity is a fundamental property of distributed systems, as the different distributed entities have to be able to communicate in order to cooperate and to coordinate different sub tasks. Ubiquitous computing has a particularly strong demand towards connectivity [HMNS01], which is motivated by the necessity of seamlessly integrating the manifold heterogeneous devices that are expected to populate ubiquitous computing environments. And according to Lou Gerstner,

former Chairman and CEO of IBM, we will soon "see this hyper-extended networked world – made up of a trillion interconnected intelligent devices" [Ger98].

3.3.8. Ad Hoc Wireless Communication

Wireless communication is a prerequisite of ubiquitous computing, providing the technical means of localized interconnectivity between mobile entities and of access to infrastructure-based communication networks. Already Weiser stated the need of "cheap wireless networks everywhere" [Wei91]. Depending on the particular purpose, both infrastructure-based and localized *ad hoc* wireless communication is required, to enable the communication and cooperation between mobile entities and remote networked services with the help of wireless network gateways, and to stimulate direct local interaction among (mobile and stationary) ubiquitous computing entities and smart objects themselves.

3.3.9. Open-World Assumption

In order to enable interoperability and connectivity across platforms and boundaries in the face of a high degree of hardware and software diversity, common *open standards* are a crucial requirement [HMNS01]. Ubiquitous computing encourages open systems based on an *open-world assumption*. Or as Lou Gerstner described his vision of interoperability during a keynote speech at CeBIT '98 in Hanover, Germany [Ger98]: "Everybody's software, running on everybody's hardware, over everybody's network". The different decentralized smart entities "cooperate in an open mutual community", by establishing a "dynamic network of relationships" [HMNS01].

3.3.10. Volatility of Cooperative Relationships and Topologies

In ubiquitous computing environments, the nature of cooperative relationships and partnerships among devices is highly *volatile*. This is a direct consequence of mobility, as mobile devices may frequently and without notice enter and leave the range of direct/indirect communication with other potential interaction partners. Another reason is the inherently high dynamics of ubiquitous computing systems, which results in frequent changes of device topologies and resource availability. The number of entities involved in providing a service changes over time, depending on the availability and/or physical presence of those entities, their internal states, the situation of the user (change in intention, required quality of service, etc.). The result is an *ad hoc composition* [GDL+04] of services, applications, and devices at runtime. Further, communications and interactions between smart objects are liable to be initiated (and ended) spontaneously [Mat04].

3.3.11. User-Centricity

Ubiquitous computing does not aim at being separated from the real world, but it instead aspires to mold the familiar everyday environment of the user by means of unobtrusive technology, with the goal of improving the user's quality of life [Mat03].

Or in Marc Weiser's words, ubiquitous computing provides computation that "works primarily in the background where it may not even be noticed", unobtrusively supporting people in everyday life situation, but at the same time leaving "you feeling as though you did it yourself" [Wei93b]. So as ubiquitous computing provides ubiquitous access to information, communication, and computation by having users employ many different mobile, stationary and embedded computers over the course of the day, ubiquitous computing focuses on mobile people and not just on mobile computers [SAW94]. Therefore ubiquitous computing can be considered a vision of *human-centered technology* [Mat03]. In the long run, ubiquitous computing systems may even be considered the spearhead of a new generation of *human-centered systems* [Der02].

3.3.12. Ubiquitous Computing Environments and Clients

In the remainder of the document, we use the term *ubiquitous computing environment* to refer to (1) a physical environment which is populated with various entities (physical artifacts and devices, mobile or stationary, large or small) that have been augmented with ubiquitous computing technology, and to (2) the sum of services and applications provided by these augmented entities. In accordance with Satyanarayanan [Sat01], we further assume that each user is immersed in a personal computing space that accompanies him/her everywhere and mediates all interactions with the ubiquitous computing elements in his/her surroundings (e.g., with nearby smart objects). This personal computing space is likely to be implemented on a body-worn or handheld computer (or a collection of these acting as a single entity). We refer to this entity as a *mobile user device*, or simply *mobile device*. Sometimes we may also refer to this entity as *client*, as suggested by Satyanarayanan, though the nature of many of its interactions may be peer-to-peer rather than strictly client/server.

4. Dependability in Ubiquitous Computing

As we saw earlier in Chapter 3, ubiquitous computing systems are significantly different from traditional distributed computing systems. Within a ubiquitous computing environment saturated with computing and communication capability, we find a potentially large number of autonomous, heterogeneous computerized entities that are capable of interacting spontaneously or of changing cooperative relationships at random. If smart objects or devices are mobile, they have to adapt to changes in their surrounding environment, which includes changes in the number of available resources, services, or communication bandwidth. Cooperating devices may be disconnected or leave the area of influence of other devices for an undetermined amount of time, possibly never to be seen again by their former cooperative partners. In case a device is portable, it may further be subject to inadvertent temporary absence or permanent loss. And due to the user-centric nature of ubiquitous computing applications, there are situations in which it is not possible to simply abstract from the identity or location of a networked device or service. In traditional system-centered distributed computing systems, a backup software process or a standby hardware component can usually be executed on a different computer in the network to compensate for the failure of the primary process or component. An individual person in a ubiquitous computing environment, however, relies on the concrete instances of tangible devices that are in his or her possession or physically present in his or her immediate vicinity in order to interact with the surrounding smart environment at the current location.

In the following, we first describe typical faults that occur in ubiquitous computing environments. Then we briefly review the four traditional methods that can be applied to protect the dependability of a computing system in the context of ubiquitous computing.

4.1. Faults in Ubiquitous Computing Environments

Traditionally, faults are distinguished as *hardware faults* in case a physical hardware component ceases to function according to specification, or as *software faults* which are the result of incorrect design or erroneous programming ("bugs") caused by human errors in the software. A hardware fault can be further classified as either a *crash fault* (the component halts or loses its internal state), *omission fault* (the component does not respond to some inputs), *timing fault* (the component responds too early or too late), or *Byzantine fault* (the component behaves in a totally random manner) [Jal94]. In ubiquitous computing environments, hardware faults are liable to occur frequently, as the user is expected to be surrounded by a

potentially large number of unreliable low-cost computerized devices and artifacts (see also Sect. 3.3: pervasiveness and ubiquity). Portability further increases the likelihood of hardware failures due to physical damage, as devices and objects are liable to suffer from percussions during transport.

In ubiquitous computing systems, there are additional sources for faults to occur with regard to a smart object or device, which are not considered in conventional distributed systems:

4.1.1. Ad Hoc Communication Faults

An ad hoc communication fault may lead to the inability of establishing or to the unintentional termination of ad hoc communication links with other entities in communication range. Such faults can be caused by (1) incompatible communication technologies/interfaces/software versions, or by (2) the loss of (wireless) connectivity due to interference, object mobility, the spontaneous turning-off of devices (e.g., performed manually by the user or automatically by the object itself when switching into standby-mode for energy saving), or physical damage.

4.1.2. Coordination and Synchronization Faults

A coordination and synchronization fault may effect the disruption or corruption of cooperative relationships between devices or between devices and external resources. This is the case if a cooperation partner or a used resource (1) spontaneously loses the willingness (e.g., user manually terminates cooperation) or ability (e.g., due to physical damage or a lack of resources) to cooperate and therefore terminates established cooperative relationships without further notice, (2) becomes temporarily or permanently unavailable (e.g., by moving out of ad hoc or infrastructure-based communication range, by changing a previously used address or identifier, or due to physical damage or destruction), or (3) partially or completely loses its internal state (e.g., due to energy-loss, software error and restart, operating system error and reboot, etc.) and thus gets out-of-sync with regard to previously established cooperative or usage relationships.

4.1.3. Inaccessibility Faults

An inaccessibility fault may lead to the unanticipated physical unavailability of a user device or user interface device. Such faults occur in case a user unintentionally lacks the physical means to locally access services or particular device functionality (which are otherwise fully operational and unaffected by faults).

Ad hoc communication faults and coordination and synchronization faults typically affect the interaction between smart objects and devices and lead to *technical* or *system-oriented* failures. They are largely a result of the high system dynamics and volatility of inter-device-relationships in ubiquitous computing environments.

However, *inaccessibility faults* describe a novel non-technical, *user-oriented* category of faults which, to our knowledge, so far has not been of concern in conventional distributed or mobile computing systems. Inaccessibility faults may lead to the unanticipated *inaccessibility* of devices and services on behalf of the *user* (rather than on behalf of other computerized entities), in situations where these devices or

services are essential for the user to perform a required task. *Inaccessibility faults* are intrinsic to many ubiquitous computing systems due to the user-centric character of ubiquitous computing in general. For instance, such faults are induced by the *portability* of devices and objects, which may lead to the unintentional physical absence of user devices, and by the *locality* aspect of human-computer interaction: the user needs some means of physical user interface in his or her immediate vicinity in order to be able to interact with a surrounding ubiquitous computing environment.

4.2. Fault Detection

In general, the *detection* of faults is a non-trivial task in distributed computing systems. For instance, it can be difficult to decide whether a remote computer process failed or is still up and running. The remote process, for example, may not respond due to a prolonged delay or due to the loss of asynchronous communication messages as a result of an unreliable communication system. Further, the process itself may be delayed due to overload, it may be blocked due to racing conditions, or it may have crashed and stopped. In the worst case, it may have suffered from a Byzantine failure and respond in a random, syntactically correct but semantically incorrect fashion.

Ubiquitous computing systems potentially suffer from the same kinds of faults as distributed computing systems in general do, including crash faults, omission faults, timing faults, and Byzantine faults. The higher degree of distributedness and systems dynamics in ubiquitous computing make efficient fault detection a "tough challenge" [CRC05].

4.2.1. Fault Detection in Synchronous Relationships

In the case of synchronous relationships between system components or services, ad hoc communication faults that result in the loss of ad hoc connectivity to a smart object usually can be detected in a straightforward manner. For instance, communication protocols usually signal a termination of a synchronous wireless communication link (e.g., "connection terminated" or "connection aborted"). Besides, incompatibilities concerning software drivers or communication stacks typically result in the failure of initializing a communication link or of sending/receiving data packets on the communication layer.

With regard to coordination and synchronization faults, a spontaneous termination of a synchronous cooperative relationship is difficult to detect, as the reason of the termination may also be an underlying communication problem. However, detection is possible if a termination of a relationship is explicitly signalized by negative acknowledgments indicating the withdrawal of permission (e.g., "connection refused").

4.2.2. Fault Detection in Asynchronous Relationships

The trend in distributed and ubiquitous computing is towards asynchronous and reactive systems, which cannot wait indefinitely for synchronous calls to terminate.

Asynchronous communication is usually applied in user interface systems or real-time systems, for instance [MS03]. For failure detection in asynchronous, loosely-coupled relationships between system components, the use of failure detectors based on timeout techniques such as *heartbeat messages* has been suggested [CRC05]. However, such messages are expected to add significantly to network traffic in densely populated ubiquitous computing environments. Furthermore, in case of network failures it is difficult to decide whether an entity or network failure occurred.

In general, without the availability of synchronized clocks, lossless reliable communication, and a known upper-bound on message delivery, a smart object cannot decide whether another smart object has disappeared temporarily or permanently, as the other object may reappear and respond at an arbitrarily late point in time. This problem has been reduced to the fundamental *consensus* problem [FLP85] in distributed systems (also cf. Appendix A.2.6). Similar such problems (i.e., problems where an agreement on (in)correct results or values has to be achieved among possibly faulty distributed computer entities) are synchronization, reliable communication, resource allocation, task scheduling, reconfiguration, replicated file systems, and sensor readings [BDM93]. From a system-oriented viewpoint, highly distributed ubiquitous computing systems generally constitute a more loosely coupled type of distributed systems with an inherently higher degree of system dynamics. Therefore the fundamental *consensus problem* of distributed systems in particular applies to the domain of ubiquitous computing systems, where messages can be lost and message delays are potentially unbounded in case a communication partner becomes unavailable for an arbitrary long period of time or ceases to cooperate without prior notice. The study of solutions to the consensus problem in the domain of ubiquitous computing, however, is not subject of this dissertation.

4.2.3. Detection of User-Centric Faults

The detection of inaccessibility faults is performed by the user the moment he or she discovers the absence or unavailability of a required device or service. Such user-centric faults and their detection are not of interest in traditional distributed computing, where the applied models are system-centric (i.e., they do not consider individual processes or devices that are associated with a particular user). In this dissertation, we explicitly focus on the investigation of user-centric dependability issues and solutions in ubiquitous computing settings where the user plays a significant role.

4.3. Fault Prevention

Fault prevention methods are conventionally used in the process of designing, testing, and validating applications with the goal of identifying and removing possible sources of failure before the applications are actually deployed.

In ubiquitous computing environments, the are a number of faults that cannot be prevented. First, the sheer number of mass-produced, low-priced components with limited technical dependability are prone to hardware and software faults. Further, the high level of distributedness of computerized entities, their mobility, and the dynamics and spontaneity of their interrelationships [Mat01a] add further

possibilities of incurring ad hoc communication faults or coordination faults. For instance, smart objects and devices may be turned off at any time, move out of radio range (potentially severing existing communication links), or simply run out of resources (memory, energy) because of stringent resource limitations stemming from small form factors or from reasons of cost efficiency. Mobile devices are further prone to become physically unavailable, either temporarily (e.g., if left behind unintentionally) or permanently (e.g., if broken, lost, or stolen), which cannot be prevented in practice. Fault prevention as a general method is therefore not a practical means for ensuring the dependability of ubiquitous computing systems.

Nevertheless, computer-based memory aids and reminder tools [Sil97] can to some extent help to prevent user-centered inaccessibility faults by reminding people about things they need to do, or more specifically, about smart objects or devices they need to carry along. Boriello et al. [BBH+04], for instance, developed a wristwatch-sized device that reminds users about objects they are about to leave behind unintentionally. In the process, they used passive radio frequency identification as an enabling ubiquitous computing technology for object identification.

4.4. Fault Removal and Fault Forecasting

Fault removal and *fault forecasting* methods have limited practical relevance with regard to ensuring the dependability of ubiquitous computing applications. In ubiquitous computing, the end-user typically forms an integral part of ubiquitous computing systems. The user makes use of smart devices he carries or finds in his immediate locality in order to implicitly or explicitly interact with a surrounding, omnipresent ubiquitous computing environment. The human factor introduces a significant level of uncertainty with regard to predictions: forecasting the user's future intentions, actions, or even misbehaviors in order to estimate the future incidence and the consequences of faults is a very difficult if not impossible task. Neither can such indeterministic sources of faults be anticipated a priori and removed, as it is possible with the removal of deterministic faults or "bugs" in a piece of software. This situation is aggravated by the potentially large number of low-cost computerized entities with comparably unreliable and failure-prone hardware, the volatility of their cooperative inter-device-relationships, and the unpredictability of how these computerized entities may be put to use and controlled by end-users.

4.5. Fault Tolerance Based on Different Forms of Redundancy

In ubiquitous computing environments, the occurrence of faults cannot be prevented *a priori*. However, in certain conditions, *fault-tolerance methods* make it possible to continue to provide a service that complies with the service specification even in the presence of faults. For that, these methods require some kind of redundancy of local resources to be in a position to compensate for faults (and failures induced by these faults) that are liable to appear during operation.

4.5.1. Hardware Redundancy

A classical approach for achieving fault tolerance is to substitute a failing resource with a readily-available duplicate resource at runtime, such as the *hot* or *cold* swapping of resources by using standby components (cf. Appendix A.3.2: *dynamic redundancy*). The term "duplicate" implicates that the standby component features the identical physical structure and/or logical functionality as the original component it is replacing (*homogeneous redundancy*). Further, if the additional or substitute resources that a ubiquitous computing device or application requires for tolerating a certain type of fault are location-independent, such as memory space or processing power, the device or application can also utilize resources available in a background network infrastructure as part of a conventional distributed system, assuming that infrastructure-based connectivity is available.

4.5.2. Homogeneous and Heterogeneous Redundancy

As a rule, mass-produced ubiquitous computing devices do not have reliable or redundant resources at their disposal, either for cost or size limitations. As a consequence, smart objects and portable devices in particular are liable to fail completely or become unusable in the case of faults (e.g., complete destruction due to physical damage, battery depleted, display broken, etc.). In such a situation, it is the *user's* task to exchange the device with a suitable surrogate. Either the user can replace it with another device of the same kind, or with a different type that features similar (but not identical) functionality. In the first case, if there are redundant identical devices available in the locality of the user to serve as potential surrogates, we again speak of *homogeneous* redundancy. Otherwise, if the available redundancy is represented by devices that significantly differ in terms of physical structure or logical functionality from the device that is to be replaced, then we speak of *heterogeneous* redundancy. So homogeneous redundancy is concerned with the *multitude* of *equal* resources in the ubiquitous computing environment, while heterogeneous redundancy is the result of an *abundance* in *diverse* resources (*diversity*).

4.5.3. Individuality of Devices

User devices are often personalized or customized in some way, which provides them with an *individuality*. If the user's personal data or customizations are essential for the proper functioning of the device and the services it provides, it is not sufficient to merely replace the physical device in order to achieve fault tolerance in case the original individual user device fails. Instead, either the user or the surrounding ubiquitous computing infrastructure need to provide the personal user data required for the proper functioning of the replaced device.

4.5.4. Locality Aspect of Redundancy

Given a user-centric ubiquitous computing application or device, the redundancy required for tolerating failures in components or services usually has to be recruited in the immediate vicinity of the user. For instance, unlike in a distributed server cluster or in a peer-to-peer network where the physical location of computers or

processes is of no concern, a service running on a device used by a person in a *particular location* in order to interact with the *local* environment cannot be substituted with an arbitrary device upon the occurrence of a fault. Even though other devices may logically be part of the same communication network, they will be useless to the user in case he or she depends on the presence of a physical interface in the current immediate vicinity or *locality*. With regard to ubiquitous computing systems in general, even though a mobile user far from home is expected to generate some distant interactions with sites relevant to him, the preponderance of his/her interactions will be local [Sat01].

Furthermore, as we described earlier in the characterization of ubiquitous computing systems in Sect. 3.3, ubiquitous computing encourages open systems and user-centric applications. Literally, in a ubiquitous computing environment, the single users (and their personal devices) constitute individual players that tend to follow independent, personal goals. This means that the users' devices and the processes executed on these devices are usually not or only loosely embedded into a background computing infrastructure. This naturally reduces the degree of infrastructure support to ubiquitous computing applications and thus implicitly limits the availability of remote resources as a source of redundancy needed for the realization of fault-tolerant system behavior.

Instead, the different players (i.e., devices) "cooperate in an open mutual community" by establishing a "dynamic network of relationships" [HMNS01]. In the process, the general open-world assumption of ubiquitous computing requires from individual user-focused devices and applications that they adopt a dynamic view of the world, forming temporary ad hoc alliances and cooperations with other partners as they are encountered *along the way*. Technically, the observed high system dynamics and the need of localized ad hoc interaction is mirrored in the capabilities for ad hoc wireless communication and support for mobility, which are characteristic of many devices and augmented objects found in ubiquitous computing environments as we discussed earlier. This is a further reason why user-centric ubiquitous computing devices and applications in particular require *localized* redundancy.

4.5.5. Localized Ad Hoc Redundancy

One potential solution for obtaining redundancy in the locality of a user's device it the local allocation of external resources in the direct vicinity of the user by means of a temporary *localized cooperation* with nearby devices, which we call localized ad hoc redundancy, or in short, *ad hoc redundancy*. Ad hoc redundancy can be homogeneous, exploiting the multitude of identical resources in the vicinity (e.g., using the identical display and media player of a nearby device of the same kind when the display of the user's primary device is defect), or heterogeneous, relying on the diversity of available resources. An example for heterogeneous ad hoc redundancy is a device that connects in an ad hoc fashion to a nearby smart phone via Bluetooth [BS06] in order to use it as a proxy for contacting a remote server via a cellular network-based GSM/GPRS/UMTS [GSM06, UMT06] Internet connection in case the Wireless LAN communication interface on the device itself is not functioning. Especially smart objects or user clients with significant resource limitations have to rely on additional resources from external sources (such as additional persistent memory space, computation power, remote connectivity sharing, or data forwarding

services, obtained from other smart objects in the vicinity) in case of local resource bottlenecks or component failures. By using nearby "richer" smart objects as communication proxies, it is possible to provide resource-restricted smart devices that are only capable of short-range communication with additional resources residing in the background network infrastructure. Due to the high volatility of cooperative relationships in ubiquitous computing environments, the providers of ad hoc redundancy may change frequently from the perspective of a user's individual device. Further, as a rule, a device that employs ad hoc redundancy does have no quality-of-service guarantees for services used from nearby devices, and no control over resources and data that have been outsourced in the process.

4.5.6. Quality-of-Service Redundancy

Another possibility to provide fault tolerance capabilities to individual devices is to reduce their performance and quality of service within the tolerances of the effective functional specifications and user expectations. This is possible if an application or service does not need all available resources at the same time or does not require to work them to capacity for meeting the minimum quality-of-service requirements. In this case the resources on the device are redundant with regard to the delivered quality of service as a measure. Such redundancy we call *quality-of-service redundancy*, or in short *qualitative redundancy*. Quality of service, however, is a *soft* criteria for measuring the degree of redundancy, as it is specific to a concrete task or problem. In contrast, *hard* redundancy is based on the physical or logical structure of a device (e.g., redundant physical components) or application (e.g., redundant processes).

For instance, the smart Mediacup [BGS01] we mentioned earlier could be programmed to disable its integrated chip used for temperature measurements in favor of a prolonged operability of the motion sensor in case of a low battery level, assuming that motion sensing is sufficient for inferring the activity and presence of people in a meeting room with an acceptable loss of accuracy, and therefore prioritizing motion sensing over temperature measurements.

4.5.7. Functional Redundancy

A further potential source of local redundancy is *functional redundancy*, which exploits the diversity of resources. In the context of our work, a device possesses functional redundancy if it possesses *diverse* resources that overlap with regard to a particular functionality. For instance, a battery-powered device using solar cells for light intensity measurements can also revert to using the solar cell for generating power – here the battery and the solar-cell constitute two functionally overlapping resources that yield functional redundancy with regard to energy generation. Of course, functional redundancy can also be obtained by making use of external, diverse resources of other devices. The combination of local ad hoc cooperation and resource sharing with an overlap in functionality results in functional ad hoc redundancy. Consequently, heterogeneous redundancy with regard to diverse external resources as defined earlier implies functional redundancy.

4.6. Adaptability

In situations where there is a significant mismatch between the supply and demand of a resource while the resource itself is not at fault, *adaptation* is necessary [Sat01]. In the process, adaptation concerns resources such as wireless network bandwidth, energy, computing cycles, memory, and so on. In the following, we use the term *adaptability* to refer to the capability of a system to perform adaptation of some kind.

4.6.1. Adaptability vs. Fault Tolerance

Faults tolerance enables a computing system to deal with faults that have their origin in unexpected, randomly occurring defects of system components, such as the failure of the network, of a physical part, or of a software process. Similar to the effects of a partial system failure, a *mismatch of resources* can also effectuate a system behavior which is not compliant with the specifications. For a computing system to become reliable and highly available, adaptability is an important quality in addition to the ability of tolerating faults.

4.6.2. Adaptability in Ubiquitous Computing Environments

We have seen earlier that diversification is a pronounced characteristic of ubiquitous computing systems and infrastructures, resulting in a high degree of *heterogeneity in capabilities* [HMNS01] and in an "uneven conditioning" [Sat01] of the environment. The existence of significant differences in the "smartness" of different environments (e.g., what is available in a well-equipped conference room, office, or classroom, may be more sophisticated than in other locations) is expected to persist for years or decades, if not forever [Sat01].

Furthermore, ubiquitous computing systems feature high systems dynamics and object mobility. This manifests itself in an ad hoc composition [GDL+04] of services, applications, and devices at runtime, and in the spontaneous initiation and termination of communications and interactions [Mat04]. For this reason, temporary mismatches in resources may occur even in environments that are otherwise uniformly penetrated with ubiquitous computing technology. As a consequence, adaptability is an important quality of ubiquitous computing applications.

4.6.3. Strategies for Adaptation

Satyanarayanan has identified three alternative strategies for adaptation in ubiquitous computing [Sat01]: Firstly, a client device running an application can change its behavior so that it uses less of a scarce resource. This requires quality-of-service redundancy or functional redundancy, and usually reduces the user-perceived quality, or fidelity, of an application. Odyssey [NSN+97, FS99] is an example of a system that uses this strategy. Secondly, a client can ask the environment to guarantee a certain quantity or amount of a resource. This approach is typically used by reservation-based quality-of-service systems [NCN98]. It effectively increases the supply of a scarce resource to meet the demands of a mobile device or application. Third, a device or application can suggest a corrective action to the user. If the

user acts according to this suggestion, it is likely (but not certain) that the resource supply will become adequate to meet demands.

According to Satyanarayanan, the existence of smart spaces (i.e., spaces where computing infrastructure is embedded in building infrastructure) suggests that some of the environments encountered by a user may be capable of accepting resource reservations. He also states that uneven conditioning of environments suggests that a mobile device cannot rely solely on a reservation-based strategy, as the environment may be uncooperative or resource-impoverished, forcing a device to ask applications to reduce their fidelities. Corrective actions involving the user broaden the range of possibilities for adaptation, and may be particularly useful when lowered fidelity is unacceptable.

4.6.4. Example for Adaptation

For example, assume a battery-powered device whose energy level drops below a critical threshold below which a continued operation at maximum power consumption would lead to an imminent loss of operability. A conventional means of achieving fault tolerance against a sudden energy loss is to supply the device with a redundant secondary battery to be used as a standby-unit. Upon depletion of the primary battery, the system would automatically switch to the secondary battery. However, in the absence of dedicated redundancy, other means of redundancy have to be found and harnessed in order to maintain an acceptable degree of operability that still complies with the (minimum) service requirements. Concretely, in our case, the following exemplary adaptive measures could be taken to maintain operability:

- The device dynamically reduces its workload to lower energy consumption and to prolong its operating time. For instance, the workload reduction could be performed in the following ways:
  - Temporary or permanent switching-off of non-critical services and hardware components, such as the deactivation of a wireless radio interface or of an on-device display. The switching-off can be performed automatically for components and services that are idle and not required by other services/components. If the smart object is unable of making that choice due to the absence of objective criteria, it can prompt the user to manually stop services or to deactivate hardware components that are not needed at the moment.
  - Graceful degradation of provided quality of service, e.g., by reducing the rate of communication transmissions, processing, sampling, sensing, brightness of the on-device display, etc.
  - Shifting necessary communication traffic from long range communication with remote access points towards more energy-efficient short range (ad hoc) communication (e.g., using nearby smart objects as local communication proxies).
- The device prompts the user to replace or recharge the battery, or to provide an alternative power supply (e.g., by cable).

4.7. Conclusion

Ubiquitous computing systems are significantly different from traditional distributed, mobile, and embedded computing systems. On the one side, this has technical and system-related reasons such as the particularly high degree of distributedness, pervasiveness, abundance, diversity, ad hoc interconnectivity, mobility, and portability of devices and resources, which manifests itself in an exceptionally high level of system dynamics, open system architectures, and in the pronounced volatility of cooperative relationships and device topologies. On the other side, ubiquitous computing is distinctively user-centric in character, which means that the human factor plays an essential role in the design and operation of ubiquitous computing systems and applications.

These particular characteristics lead to novel issues and challenges with regard to the dependability of ubiquitous computing systems. In contrast to established dependability models, permanent (inter-)connectivity between system entities can no longer be taken for granted, as interaction is often based on localized ad hoc communication only, which is prone to ad hoc communication faults. What is more, the openness of system architectures, the high volatility, and the strong user-centricity of ubiquitous computing systems lead to a high uncertainty about the persistence of cooperative relationships, which manifests itself in the occurrence of coordination and synchronization faults. Finally, as ubiquitous computing is set to embrace the user in his or her everyday life environment, inaccessibility faults caused by the unanticipated unavailability of user interface devices constitute a novel kind of fault that have to be considered in human-centered ubiquitous computing systems.

Since it cannot be prevented that faults of various types occur in ubiquitous computing systems, *fault-tolerance concepts* are of paramount importance to safeguard dependability. Further, in response to the high volatility of relationships and the expected fluctuations with regard to local resources, *adaptability* is a crucial capability of user-centered ubiquitous computing applications.

5. User-Centric Dependability Challenges in Ubiquitous Computing

In this chapter, we first explain the notion of *user-centric dependability*. The we discuss a number of user-centric dependability challenges in ubiquitous computing, with a focus on the areas of human-computer interaction and context-aware computing.

5.1. System-Centricity vs. User-Centricity

The widely accepted definition of *dependability* as the "trustworthiness of a computer *system* such that reliance can justifiably be placed on the service it delivers" [Lap85, Lap92a] is inherently *system-centric*: The underlying *physical model* is based on distributed physical system components, and its *logical model* considers distributed, cooperating system processes that perform the actual computing to provide specified and programmed services to the user [Jal94].

With respect to the physical model, fault-tolerant system behavior is achieved by replacing defunct components of a distributed system with redundant components to maintain operability. In the logical model, individual processes that fail have to be replaced by other processes to ensure that a computation – the service provided by the system – is continued according to specifications. In both cases, the single failing system component or process has no *individuality* of its own, but it can be replaced by redundant components that are indistinguishable from the failed one with regard to functional behavior and operation. Further, as the distributed system components are linked by physical or logical communication channels where the physical distance between single system components is of minor importance. What matters is that all components are connected to the same overall distributed system and able to communicate with each other with acceptable delay.

Likewise, conventional, system-centric dependability concepts focus on the protection of system infrastructures and installations, such as mobile phone telephony backbones, Internet domain name services, landline telephone switching networks, database and booking systems, or computer systems managing manufacturing and power plants, for instance. Thereby the end-user is mainly considered at the fringes of the computing system, monitoring or interacting with a system by means of dedicated access points, such as mobile or stationary terminals. Even in the case of mobile telephony, the main dependability challenges were traditionally seen in maintaining the operability of the underlying background infrastructure, and the robustness and effectiveness of the wireless communication channels.

Whereas cost-intensive and highly reliable mobile user devices are often employed in specialized, safety-critical systems (e.g., medical implants used in the medical

sector) and as part of heavy-duty industrial applications, such devices are usually not found in everyday life ubiquitous computing systems, where the quality and reliability of mass-produced off-the-shelf user devices and smart objects is significantly lower due to cost reasons.

Furthermore, as we have seen earlier in Section 3.3, ubiquitous computing is characterized by an unprecedented degree of *user-centricity*. The highly distributed ubiquitous computing infrastructure is no longer a distinct entity separable from the user, but instead explicitly designed to embrace and surround people in everyday life settings and situations. Consequently, the interaction of users with their surrounding smart environments, and the exploitation of the multitude of ubiquitous computing resources and devices found therein for the delivery of novel or augmented services and applications, center on the user himself as the focal point. This user-centeredness is evident in late Marc Weiser's defining mission statement of ubiquitous computing, which is to "serve the user in an unobtrusive manner in everyday life situations" [Wei93b].

As the user is moving into the center of attention, the discipline of dependability has to acknowledge the strong user-centricity of applications and services in ubiquitous computing: the user has to be seen as an integral part of the system, and the dependability concepts need to accommodate the strong interrelation between the user and the surrounding ubiquitous computing environment. The matter of *user-centric dependability* so far has received little attention in the dependability community, where the traditional system-centered viewpoint is predominating research.

In the following, we identify novel user-centric dependability issues for two fundamental areas of ubiquitous computing, human-computer-interaction and context-aware computing. For more general information about these two areas, please refer to Sections B.4 and B.5 in the appendix.

Of course, ubiquitous computing technology is not limited to providing user-centric dependability concepts only, but it can also be employed for the delivery of fault-tolerant system-centric services. For instance, in [BGV02], we describe how an existing distributed ubiquitous computing infrastructure, which features diverse ubiquitous computing devices and sensing technologies, can be used for the monitoring and logging of security-relevant activities in buildings or places, thus delivering the technical basis of a highly-distributed, system-centric *automated security auditing* system.

5.2. Dependable Human-Computer Interaction

Naturally, human-computer interaction puts the user into the center of attention: aspects of usability and user-friendliness are paramount as part of the human-computer dialogue. The importance of the usability aspect is mirrored in ISO standardization efforts for defining this term [JIMK03].

Considering the manifold publications in the field on new means for human-computer interaction, typically combined with usability studies, the strive for the development of ever new tangible user-interfaces, novel displays, embedded utilities, and networked user appliances currently dominates the scene. Dependability in the context of human-computer interaction so far mainly focused on the design and implementation of user interfaces, investigating how the lack of clarity, usability,

and erroneous design and implementation of user interfaces affect the safety or security of computer systems and applications [MS04].

5.2.1. Accessibility of User Devices and Interfaces

With the emergence of highly interactive and computerized ubiquitous computing environments, the user's dependence on computer interfaces and portable user devices has been increasing significantly even in everyday life situations. This leads to the new, user-centric type of dependability threat caused by *inaccessibility faults* (see Sect. 4.1.3): if a user requires certain devices or user interfaces to interact with a surrounding ubiquitous computing environment on a daily basis, how does he or she cope with the *unavailability* of such interfaces? The physical unavailability of a suitable user interface leads to the *inaccessibility* of data and functionality. The inaccessibility of the user interface is independent from the availability of the services providing the data and functionality as part of the ubiquitous computing environment or background computing infrastructure: even if a service itself is available and operational, it is unusable to the user if the absence of the proper user interface prevents the user from getting *access* to that service. Therefore, the *accessibility* of devices and interfaces and of the functionality they deliver is a novel critical dependability challenge in user-centric ubiquitous computing.

5.2.2. Data Persistence, Confidentiality, and Life-Cycle Management

The use of small portable devices such as smart phones and personal digital assistants aggravates the accessibility issue, as these items are especially subject to physical damage, loss, and theft. What is more, portability in particular poses the additional threats of permanent data loss and loss of data confidentiality: personal devices may permanently get out of reach for the user, or confidential information may fall into the hands of malicious third parties.

When similar dependability problems in alleviated form were first recognized in mobile computing, it was suggested to minimize the essential data kept on the portable device, to use encryption, and to employ data synchronization and replication mechanisms [FZ94]. In ubiquitous computing applications, however, it may even be wanted or necessary to carry more than the basic minimum of personal data on a portable device. The availability of ample personal data on a physical local device is a prerequisite for enabling localized services in offline or disconnected mode, or to enable applications where data is available and processed in situ, such as recorded sound or video that a user wants to edit on his device and share with other nearby physical devices.

Further, with the proliferation of computing devices and portable ubiquitous computing appliances in the form of commercial off-the-shelf products – such as mobile phones, personal digital assistants, and digital cameras – the life cycles of many such products are also becoming significantly shorter. Taking into account the higher probability of loss and damage with regard to portable devices, personal user data and meta data tend to outlive the life cycles of their physical carriers. The life cycle management of short-lived user devices and of persistent personal data and customizations constitutes an important ubiquitous computing challenge.

5.2.3. Limitations of Conventional Dependability Concepts in Human-Computer Interaction

Portable and handheld devices have become popular and ubiquitously accessible companions that support users in activities of daily living [MB03]. As they are inexpensive compared to other types of computers, users may possess more than one of these devices, each having a specialized user interface and/or functionality. Furthermore, handhelds are expected to increasingly serve as a personalized, single point of access for controlling electronic equipment and devices in the user's environment [NMH+02]. Consequently, in emerging ubiquitous computing scenarios, there will be situations where the user relies on handhelds to provide customized device-specific functionality and access to arbitrary personal data in a spontaneous fashion without much ado and delay. Data synchronization and replication mechanisms are apt to prevent data loss, but they do not protect the loss of the user-interface and interaction device itself which would incapacitate the user from interacting with surrounding smart environments. In ubiquitous computing scenarios where the user depends on mobile user devices to perform everyday life chores and tasks, such a loss of interaction capability and control may lead to severe discomfort or safety risks. Besides, encryption is a valid option to protect data confidentiality, however it may not be a practical solution for the user in everyday living as frequent access to personal data would require frequent and cumbersome authorization, and because low-cost resource-limited devices may even only have limited support for cryptographic functionality for cost-saving reasons, resulting in slow or insecure encryption. This calls for novel user-centric dependability concepts that ensure the accessibility of mobile user devices needed for the interaction with smart environments, and which protect the confidentiality of personal data with regard to portable devices that are particularly liable to loss and theft.

5.3. Dependable Context-Aware Computing

In general, by the application of context awareness, a single device can exhibit different semantics in different contexts, and thus adapt itself dynamically and non-intrusively to a user's circumstances. Context awareness is therefore considered a *user-centric* view of computing [Nel98].

5.3.1. Dependability of Indirect Context-Awareness Systems

If context awareness is provided indirectly to devices by means of background infrastructures, the dependability of these infrastructure-based services can be protected by means of conventional dependability mechanisms, such as primary site or active replication strategies (see Appendix A.4.2). Mobile devices that rely on such *indirect* context information require connectivity to be able to communicate with the infrastructure-based services. Hence the dependability of the indirect context-awareness is a system-centric dependability challenge of the background infrastructure and of the communication link.

5.3.2. Dependability of Direct Context-Awareness Systems

Direct context refers to context information that is retrieved autonomously in the respective location where it matters [GSB02]). In dynamic ubiquitous computing environments featuring specifically equipped smart objects and mobile devices – from the perspective of the user – *direct* context-awareness is much more relevant than indirect context-awareness. This is because mobile and portable devices are subject to frequent context changes and have to be able to adapt to a high volatility of cooperative inter-device-relationships and locally available resources. Therefore mobile devices particularly benefit from direct context information, as it reduces their dependence from remote services and networked background infrastructures, fostering autonomous and disconnected local operation.

For some applications the semantic higher-level context of the user may be of interest, such as the user's current intention or activity. On the fundamental level of computing, however, the physical, low-level context of a (mobile) device is most relevant. This context consists of physical environmental properties, resources, and services found in the sphere of direct interaction of the device with its surrounding physical environment. The latter we also refer to as the vicinity or *locality* of the device. Low-level context potentially provides localized access to additional redundant resources that may be tapped for fault-tolerant computing. In particular, the knowledge of higher-level context itself can only be gained through the processing and fusion of lower-level or raw context data, as the low-level context provides the basic, technical means to explore, sense, and interact with the environment.

Dependability is an issue on two different levels of abstraction with regard to direct context-aware computing: the process for obtaining and inferring context should be rendered dependable and fault-tolerant, and the dependability of ubiquitous computing applications themselves should be improved by exploiting context information and local external resources.

Part II.

Dependability in Human-Computer-Interaction

6. Concepts for Dependable Human-Computer-Interaction

In the following, we further motivate the importance of ensuring the accessibility of services and devices in ubiquitous computing for dependable human-computer-interaction. Then we introduce the two concepts of *Input/Output Diversification* and *Instant Personalization*, which address the accessibility issue.

6.1. Accessibility of Devices and Services as a Fundamental Challenge

In general, a user requires some kind of physical user interface – which we call *user interface device* (UID) in the following – to be in a position to explicitly interact with his surrounding ubiquitous computing environment. The UID can either be provided as part of the local infrastructure (such as a public Internet terminal, a personal computer at home, or a smart shopping cart), or be carried by the user (such as a notebook computer, personal digital assistant, digital camera, or mobile phone). The UID can further either be a public device devoid of personalization information, or personally owned and customized by the user.

6.1.1. Accessibility as a User-Centric Dependability Challenge

From the perspective of the individual user, the dependability of the interaction with his or her surrounding smart ubiquitous computing environment is violated when the UID that the user customarily relies on is unavailable for unforeseen disturbances the very moment its services are required for accomplishing a certain purpose (inaccessibility fault). As a result, the user is denied the *accessibility* of UID-specific functionality, such as access to personal information (e.g., personal calendar, phone book, passwords and electronic passkeys, etc.) or particular services (e.g., making calls, writing a letter, taking a photo, etc.). The unanticipated unavailability of the UID in general is the result of two main reasons: the *failure* or the *physical unavailability* of the UID. These conditions can further be characterized as temporary or permanent according to their temporal duration. The unavailability is *temporary* if it resolves itself or if it can be actively eliminated by the user after a certain period of time that outlasts the period in which the user would have required the service of the UID. This corresponds to a temporary disturbance that prevents the user from utilizing a UID for accomplishing a particular purpose. A permanent disturbance does not resolve itself over time and cannot be removed by the user in the current situation. Instead, it requires professional assistance and/or the future replacement of some or all components of the UID.

6.1.2. Functional Unavailability

The *failure* of a UID (e.g., caused by hardware, software, communication, or coordination faults) makes it unfit for service and renders it unusable, which leads to its *functional* unavailability. Failures can have variable causes, such as a random technical defect during operation due to hardware, software, or configuration faults, physical damage because of an extraneous influence (e.g., a device is dropped and breaks upon impact, or water seeps into a mobile phone and short-circuits it), and temporary shortage of resources required for computing such as lack of memory space, communication bandwidth, connectivity, or energy. A failure due to loss of energy can be the result of an electrical power outage or of a depleted battery, for instance.

6.1.3. Physical Unavailability

The unexpected physical unavailability of a UID can be caused by accidental loss, intentional theft, or temporary displacement. *Temporary displacement* describes the situation where the UID is left behind in a safe place where the user can retrieve it at a later point in time. The temporary displacement therefore can be considered a *private* dependability issue. In contrast, *intentional theft* or *accidental loss* constitute *public* dependability threats since in these cases the UID leaves the sphere of control and possession of the user: the UID along with potential user data is liable to be permanently lost and unavailable. In addition, this may result in the loss of the confidentiality of personal user data (e.g., personal addresses, user passwords, electronic door keys, credit card information, etc.) and in the potential abuse thereof.

6.2. Redundancy Through Diversity and Multitude of User Interfaces

We have seen that the two fundamental issues of the user-centric accessibility challenge are the functional and physical unavailability of the user's technical medium of interaction, which we called UID. Concepts aiming at increasing the dependability of human-computer interaction in ubiquitous computing consequently have to provide the user with means of overcoming the unavailability of the UIDs he requires to maintain accessibility of personal information and services.

6.2.1. Tolerating Functional and Physical Unavailability

Ideally, the physical or functional unavailability of a UID should be averted in the first place, which corresponds to fault-avoidance according to the established conventional dependability terminology. However, this is not a realistic option for highly dynamic ubiquitous computing settings where potentially a large number of low-cost computerized devices spontaneously interact. Technical devices are liable to break and cease functioning because of technical failures and environmental influences. Further, UIDs are often represented by handheld or portable devices, such as mobile phones, PDAs, or notebook computers. It is the nature of such

mobile and portable devices that they are subject to being left behind, getting lost or stolen.

Since we naturally cannot prevent an unanticipated unavailability of an *individual* UID, the user has to be put into a position to cope with hardware, software, communication, coordination, or inaccessibility faults leading to the failure or physical unavailability of UIDs as part of the human-computer interaction. Since the user requires some kind of interaction device or interface, he or she needs to be supplied with an adequate replacement or *substitute*, which provides the user with the redundancy required for achieving fault tolerance. Based on a ubiquitous computing environment, the nature of redundancy concerning potential substitute UIDs is twofold: either we can exploit the *diversity* or the *multitude* of devices found in the immediate *locality* of the user.

6.2.2. Motivating Example

In the following we give a basic and practical example that clarifies the use of diversity and multitude of resources for increasing the availability of a UID, and for preserving the accessibility of needed personal data (phone number) and functionality (making a phone call). Imagine a user named John who needs to make an urgent private phone call but who discovers that his mobile phone is (physically or functionally) unavailable. Firstly, if John also possesses a PDA with Wireless LAN connectivity, he could try and make the call by means of Voice-over-IP Internet telephony software. Alternatively, he could look for the nearest public phone booth instead. This is an example for the use of two *diverse* user interfaces in case of the inaccessibility of the preferred primary user interface (his mobile phone). However, while John may be able to easily retrieve the phone number from his the personal PDA for making his call, he may not be able to use the public phone if he does not know the phone number by heart, given that no phone directory is available or that the number he needs is confidential and not included in the public directory. Secondly, John may possess *multiple* mobile phones, such as one for business use and one for making private calls. Given that he left his business phone at home but still has his personal phone, and assuming that he regularly synchronizes his personal contacts with both phones, he is in a position to use his private mobile phone instead: for the specific purpose, given the circumstances that both phones contain the required personal user data required for making the call, both devices are *interchangeable* with regard to functionality.

6.3. Input/Output Diversification

In the following we introduce the concept of *input/output diversification*. We show how it can be applied to address the accessibility challenge of human-computer interaction in ubiquitous computing by providing the user with redundant means of interaction that accommodate the user's varying needs in different situations and locations.

6.3.1. Diversity of User Interfaces and Capabilities

A natural solution to overcome the physical or functional unavailability of a preferential UID is for the user to look for *alternative*, *diverse* means of interaction. In a ubiquitous computing environment, we often find a variety of diversified UIDs that suit specific groups of users for specific purposes [HMNS01]. The diversification of these devices is mirrored in differences of hardware and software, which also manifests itself in heterogeneous device capabilities [HMNS01], such as diverse communication protocols, operating systems, application software, form factors, and input/output capabilities. Very often we find an "uneven conditioning" [Sat01] of the environment, as the capabilities and resources of different devices and computerized entities may significantly vary (e.g., comparing a full-fledged PDA with a basic passive RFID tag).

Even if the capabilities and properties of different UIDs vary significantly, they often have basic features in common. For instance, a laptop and a mobile smart phone both provide the user with the ability to write textual notes or to browse the Web. Diverse UIDs therefore can be used to solve the same task, but they do so with varying degrees of user-friendliness and aptitude, as each device has its specialized user interface and optimized functionality [MB03].

6.3.2. The Concept of Input/Output Diversification

The process of exploiting the diversity of input and output devices for the diversification of a user interface for the interaction with a smart ubiquitous computing environment we call *Input/Output diversification* (in short: I/O diversification). Enabling users to employ diverse UIDs for interaction provides them with (1) diverse means of input for obtaining information from the environment, and (2) diverse means of output. Both capabilities are necessary for the user to be in a position to control services and manipulate objects of the surrounding ubiquitous computing environment.

We have investigated the diversification of the interaction between a person and a surrounding ubiquitous computing infrastructure based on a practical setting. We focused on the university campus as an application environment that provides a very interesting setting for applying ideas from the field of ubiquitous computing [Wei91, Wei98]. In a campus environment, a substantial number of users share a large amount of their information needs. These needs include information about schedules, locations of class rooms, lectures, assignments, lab equipment, presentations, seminars, sports events, student ads, etc. Much of this information is directly related to objects, places, and people that are situated within the campus environment. The exhaustive installation of wireless computing facilities such as Wireless LAN, together with small handheld devices and technologies for detecting objects or locations, make it possible to satisfy the information needs of users in a campus environment in new ways.

6.3.3. Case Study: I/O Diversification in the ETHOC System

As a result of our investigation of the concept of I/O diversification, we developed and implemented the ETHOC system. "ETHOC" is an acronym for "EveryThing

Has Online Content". The ETHOC system enables users to attach virtual information and online functionality to printed material, and it makes this information and functionality available to other users via a diverse set of UIDs. The system consists of an ETHOC server application residing in the background infrastructure or in the Internet, and of software interfaces for a set of diverse ETHOC client UIDs. The ETHOC background system supports the creation, administration, and intermediation of online resources. To information providers, it offers a Web-based author portal for generating unique IDs that can be printed as barcodes and for associating online content and actions to printed material. To users it offers simple means of input/output to interact with virtual representations of printed documents using a variety of diverse UIDs, thus implementing the concept of I/O diversification: WAP-enabled mobile phones, personal digital assistants, notebook computers, and public web terminals. A more detailed account on the design, implementation, and evaluation of the ETHOC system is given in Chapter 7.

Barcodes as Visual Identification Technology

For the user to select and identify an object for interaction with his UID, we used one-dimensional barcodes as a visual identification technology (also cf. Section B.3.4). We have augmented each barcode with a textual representation of its identifier and with an ETHOC label - the resulting marker we call an *ETHOC code*. To access the information linked to a document that was tagged with an ETHOC code, the user can utilize a barcode-reader connected to his UID to scan the ETHOC code. Alternatively, if no barcode reader is available, the user can read the barcode identifier and manually enter it into the UID to select the corresponding document for interaction. Each ETHOC client caches the ETHOC codes it obtained and forwards this information to the ETHOC server once connectivity is established. Based on the type of online information attached to a document, the ETHOC client allows the user to retrieve additional information provided earlier during the authoring phase by the author of the document.

Customized Online Functionality

The ETHOC client enables the user to access certain customized online services related to a document, which include the adding of calendar entries to his electronic calendar, the registering for a course or event, participating in an online discussion forum about the content of the document, or subscribing to a notification service which informs the user about latest changes to the online content. Whenever an ETHOC code is accessed, the ETHOC system keeps a personal access history for each user. This allows the user to switch between different UIDs while maintaining access to information previously scanned with other UIDs. To address the issue of uneven conditioning and heterogeneity of UIDs, the ETHOC server uses independent modules (servlets) which perform the adaption of content and content representation to accommodate the capabilities and limitations of the various client UIDs.

Support for Arbitrary Objects

The ETHOC system can easily be extended to support arbitrary objects and identification mechanisms, because the underlying system architecture is modular and extensible. On the client side, it is possible to use arbitrary identification technologies to retrieve the ETHOC code attached to a physical object by implementing stubs for the corresponding identification hardware. On the server side, the modular architecture allows for a simple integration of additional services. In our case, barcodes had the advantage that they can be produced without extra costs and without special hardware, that users can easily integrate them with printed documents during the authoring process, and that barcode reader devices for mobile and stationary use are readily available off-the-shelf at moderate costs.

6.3.4. Motivating Scenario

The following scenario describes – from the user's perspective – how a more dependable and flexible interaction with augmented everyday objects can be achieved by exploiting the diversity of input/output devices. Based on the ETHOC [RB03] system where every thing has online content, it shows how a variety of complementary devices can be used to interact with environments where electronic information is linked with physical objects that serve as physical hyperlinks into topically related information spaces. Firstly, by providing diverse means of information input/output, the user is free to pick the device of his choice, that is the device which appears most suitable for the intended interaction in the user's particular situation. Secondly, by allowing the interchanging use of different clients that share the same access history, a *diversification of the information access* is achieved: individual devices may be lost or broken, but the information that has been retrieved once remains accessible. Further, having diverse networked devices at one's disposal, these devices can be used independently from each other, which reduces the dependence on a single device or technology. This way, by exploiting redundancy implicitly provided by diverse devices with overlapping functionality, hardware faults or inaccessibility faults with regard to individual devices or services can be tolerated.

> Bob has just graduated and decides to sell some of his furniture via the student bulletin board before he moves out. He prepares the ad with his favorite word-processing program, including a short description of the items on sale, his address, and a date for the sale. He uses the author interface of the ETHOC web interface to get an ETHOC barcode – formatted as a GIF image – which he inserts into his ad. Bob enters his contact information and the date and time for viewing the items on sale. Additionally, he takes some pictures of the items and uploads them to the ETHOC system to be added to the virtual representation of the ad. Finally, he uploads the ad, prints it, and sticks it to the student bulletin board. The ETHOC barcode now provides a physical hyperlink to an augmented online representation of his ad maintained on the ETHOC server residing in the Internet.
>
> Meanwhile, in Alice's introductory computer science class, lecture handouts and exercise sheets are distributed. They contain ETHOC bar-

codes that are linked to a newsgroup for discussing the exercises, to a source code skeleton needed to solve the programming assignment, and to survey articles relevant to today's class. Alice scans the codes with a tiny barcode reader that is attached to her mobile phone and the links are stored in her personal online history.

After the lecture, a poster announcing an interesting talk attracts her attention. She scans the attached code with her phone. The talk is added to her personal history and a WML page optimized for the web browser of her phone is instantly displayed, shortly describing the speaker's biography as well as directions to the lecture hall. It also indicates that the location of the talk has been changed. Clicking on an item labeled "appointment" inserts an entry for the talk into her mobile phone's calendar. A little later she spots a posting on the student bulletin board announcing various things for sale. She decides to study it later and quickly scans the ETHOC code of the ad to store it in her access history.

At home she connects her laptop to the Internet and – using the ETHOC Java client – looks at the items she has scanned today. She first selects the sale entry and looks at the photos of the items. Using the contact information, she calls Bob for an item she is interested in and verifies the date and time for the sale. Later, she selects the exercise entry on the ETHOC client, upon which the source code skeleton for the programming assignment is automatically downloaded. Following the resource links, she finds two papers that discuss an aspect of today's talk in detail. At the exercise newsgroup she posts a question about a particular topic she did not understand during the lecture.

The next day she is having a coffee with friends at the cafeteria. Talking about Bob's sale, her friends also wish to take a look at the furniture. Since Alice has left her laptop in a locker, she looks up the ETHOC code of Bob's ad on her mobile phone and uses the public Web terminal of the cafeteria to display the ad together with the photos on the terminal screen. As her friends are also interested in some of Bob's items, she wirelessly transmits the appointment for the sale, which includes the ETHOC code of the ad, from her phone to her friends phones, when suddenly the alarm on her own mobile phone goes off. It is the reminder for the talk that she would have missed otherwise. She quickly verifies on her mobile that the location has not changed in the meantime by revisiting the ETHOC page of the event before leaving the cafeteria to attend the talk.

6.4. Instant Personalization and Temporary Ownership

In this section we introduce the concept of *instant personalization and temporary ownership*. We motivate how the concept contributes to solve the accessibility challenge of human-computer interaction in ubiquitous computing environments

by enabling users to easily personalize arbitrary handheld devices on-demand and to temporarily share the usage of personal mobile devices that belong to different users.

6.4.1. Ubiquity of Mobile and Personal User Devices

As Weiser foresaw "a world of fully connected devices, with cheap wireless networks everywhere" [Wei91], the availability of ever smaller and cheaper computing and communication technologies [HMNS01] gradually enables a large-scale, exhaustive deployment of objects and devices enhanced with ubiquitous computing technology, making them truly *ubiquitous*. Thus the term *ubiquity* describes the characteristic of "anywhere" computing, referring to the property that ubiquitous computing technology does not just occur in individual places, but instead in abundant quantities "throughout the physical environment", with a particular stress on *everyday life situations*, as envisioned by Weiser [Wei93b]. In practice, ubiquity is a result of the *multitude* and *abundance* of smart objects and devices that are found in the user's environment. A prominent example for ubiquity is the massive proliferation of mobile phones (and of mobile phone access points), which – according to Gartner – were sold in quantities surpassing six hundred of millions of units worldwide in 2004 [Gar05a], with sales expected to exceed 730 million units in 2005. As a consequence of this development, a mobile phone market penetration in excess of 100% is expected for Western Europe as a whole by 2007 [Ana05]. Further, Informa says that the global wireless market is about to hit 2 billion subscribers by end of 2005 [inf05], after having reached 1.5 billion in mid 2004 [inf04], which results in a global mobile penetration of about 30%. A similarly rapid growth can be observed at the market for personal digital assistants (PDAs): driven by wireless e-mail applications, worldwide PDA shipments increased 25 percent to 3.4 million in the first quarter of 2005 compared with a year ago, with a 84 percent growth in shipments to 1.3 million units in Western Europe [Gar05b, KCMT05].

6.4.2. Increasing Dependence on Mobile User Devices

As we have seen earlier, handheld devices in particular have become inexpensive and popular companions that support activities of daily living, forming ubiquitous computing tools in everyday life situations. Today, an estimated 30 million Personal Digital Assistants (PDAs) and about 1.3 billion mobile phone devices are in use worldwide, with sales being expected to increase considerably within the next years [MB03]. Thanks to their portability and ease of use, mobile user devices enable convenient ubiquitous access to personal user data in situations where bulkier devices such as laptop and desktop computers are inappropriate. Being small and lightweight everyday companions that fit into a pocket, they can be employed while being on the move, often even supporting hands-free operation.

As we gradually depend more and more on the assistance of mobile user devices at work and in daily life, the reliability and availability of these devices and of the particular services they provide us with becomes a crucial issue. Firstly, a handheld device at hand may not function properly due to a technical defect or because of an empty battery, for example. This problem is aggravated by the fact that low-cost, mass-produced handheld devices are more prone to suffer from

hardware failures than higher-priced quality products used for more demanding professional activities. Secondly, a personal device may be physically unavailable, either only temporarily when the user has forgotten to take his or her device along, or permanently in case the handheld is lost or stolen.

6.4.3. Evolution and Inversion of Device Characteristics

Over the last decades, the characteristics of everyday life communication and computer devices and their accessibility as UIDs have changed tremendously with regard to mobility, portability, and their degree of personalization.

Figure 6.1 illustrates the evolution and inversion of characteristics of UIDs in terms of scope of deployment (public/many users, semi-public/restricted user, private/single user), degree of personalization (impersonal/stateless vs. personal/stateful), degree of mobility (stationary, limited portability, full mobility), and devices-per-user ratio (one device per many users/any user has access, one device per some users/restricted user group, and many devices per individual user/exclusive personal use). In the following, we discuss this process in more detail.

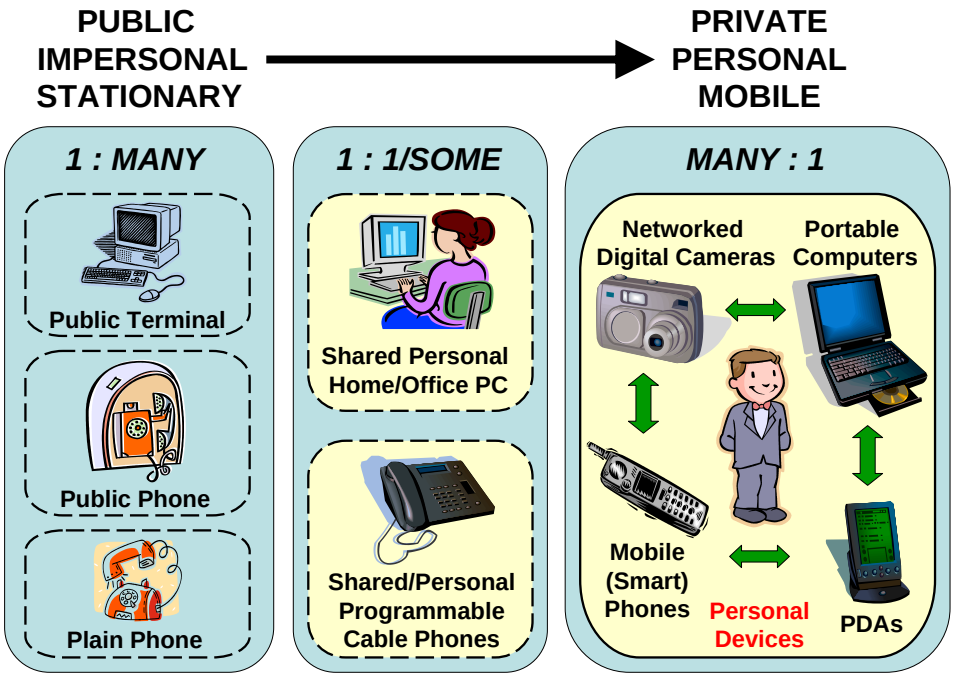

Figure 6.1.: Evolution and inversion of characteristics of user devices in terms of scope of deployment, degree of personalization, mobility, and devices-per-user ratio

Impersonal Stationary Devices

In the beginning, *impersonal stationary devices* predominated in both public and private spaces, such as public computer terminals, public pay phones, or plain old stationary telephones. Due to their generic functionality, these devices have the advantage of simple deployment, setup, and configuration. In addition, they are suitable to be shared among large, open user groups. However, the generality of

these devices also implies a lack of personalization capabilities, which limits the efficiency and ease of use on behalf of the end-user. Further, their lack of mobility and the usually sparse distribution density (e.g., only one plain phone per building, one public pay phone per city quarter, etc.) severely limits the accessibility of these UIDs: one device has to serve many people.

Personal Stationary Devices

With the availability of *personal stationary devices* in larger numbers, such as personal programmable cable phones or personal computers at home and in the office, single users or small user groups can be granted their own individual device of a kind. This enables a more uniform distribution of UIDs, which improved both the accessibility and convenience of these devices and of the services they deliver. For instance, several rooms of a building or office could be equipped with a phone and/or personal computer. Moreover, the support for personalization and customization increases the ease of use and efficiency. However, this also has the effect that the access of personalizable user devices is restricted to smaller, closed user groups: one device serves a few people only. Thus the original quality of sharing and pooling of devices that was exhibited by impersonal (stationary) devices is limited in the process.

Personal Mobile Devices

Recently, with the emergence of ubiquitous computing, *personal mobile devices* (or personal mobile UIDs) have been rapidly proliferating and becoming everyday life commodities, as the ubiquity of personal digital assistants (PDAs) or mobile phones today shows, for example. As a result, a single user typically personally owns multiple, highly customized and personalized devices, such as one personal mobile phone, one PDA, one networked digital camera, and one computer notebook. Owing to the portability and strong user-centricity of these devices, their accessibility and ease of use is greatly improved, enabling a convenient anywhere, anytime usability. However, due to the high degree of personalization, customization, and personal ownership, personal mobile devices are mostly restricted to an exclusive personal use by the respective owner. This strongly limits the sharing and pooling capabilities of these devices.

For the user, a particular consequence of using personal mobile devices is the increased complexity of personal data management. The user himself has to take care of the synchronization, persistence, and confidentiality of his personal user data that is liable to be distributed over his various personal devices. This situation is aggravated as commodity user devices tend to have an ever shorter life-cycle. This implies that the user also has to perform the task of migrating data from one personal device to its successor, which can be a cumbersome and difficult procedure in case different hardware and software standards are involved, for instance.

6.4.4. Exclusive Personal Use and Ownership as Barriers to Accessibility

An important observation is that the strong user-centricity and individuality of personal user devices provide a barrier to the sharing and pooling of these devices.

A user may refuse to lend a personalized device to another person for different reasons. On the one hand, the user may want to protect the *confidentiality* of personal data, such as personal photos, contacts, correspondence, spreadsheets, passwords, etc., which are permanently stored on his devices. On the other hand, the user may want to prevent the *misuse* of his devices. Potential misuse is liable to occur with regard to *illegitimate manipulation* of personal user data (e.g., the editing or deletion of contacts, appointments, spreadsheets, etc.), *unauthorized or excessive use of billable services* (e.g., unauthorized payments using the electronic wallet/credit card information, or excessive use of a mobile phone subscription, etc.), or with respect to *identity theft* (e.g., the borrower makes illegitimate phone/VoIP calls, or sends unauthorized instant messages/e-mails/faxes in the name of the owner of the device).

For these reasons, an abundance of personal mobile devices in a ubiquitous computing environment does not automatically lead to a high availability of such devices to the individual user, or to a high accessibility of the device-specific functionality these devices deliver. Instead, the increasing degree of personalization and customization of mobile and portable user devices constitutes a serious impediment to the accessibility of UIDs, as it restricts access to individuals and opposes a spontaneous sharing with other, possibly unknown persons.

6.4.5. The Concept of Instant Personalization and Temporary Ownership

The particular strength of mobile and handheld devices is their *portability* and *special device-specific functionality*, which enable the user to accomplish particular tasks anytime and anywhere, and the advanced personalization and customization capabilities that enhance the ease of use and efficiency in the process.

The personal meta data, individual preferences, device and application settings, customizations, user names and passwords, and so on, define the *individuality* of a mobile user device. As we have seen earlier, this individuality of mobile user devices is the main cause for an exclusive personal use and permanent ownership, which usually makes the devices inaccessible to persons other than their respective owners. In response to the user's increasing dependence on mobile and portable user devices for the interaction with ubiquitous computing environments, we propose the concept of *instant personalization and temporary ownership of mobile devices*, which addresses these issues. Instant personalization makes personal user devices and handhelds interchangeable and shareable, while preserving the intrinsic advantages of personalized mobile user devices.

As a proof of concept, we developed and prototypically implemented a system for the instant personalization of personal digital assistants (PDAs). The system enables the user to make the transition from requiring a specific *individual* PDA to utilizing *any* instance of the same type of device at hand. This significantly raises the degree of redundancy of devices accessible to the user from one to a potentially unlimited number of devices. The system that we prototypically implemented further provides support for periodic data backup, remote data recovery, and data confidentiality protection when devices are lost or stolen. The description of our prototype system is presented in Chapter 8, together with a more detailed discussion of important design aspects and challenges.

6.4.6. Motivating Scenario

This scenario demonstrates how the concept of instant personalization of portable devices reduces the user's dependence on individual personal devices by making them interchangeable while preserving personalization: instant personalization empowers people to use *any* mobile device of a kind rather than depending on a particular one. Thus the user still has access to customized device-specific functionality and personal user data required for interaction with ubiquitous computing environments even when a preferred personal user device such as a personal PDA or mobile phone becomes temporarily unavailable (e.g., the device is left behind or its battery is depleted) or permanently (e.g., in case of loss, theft, or destruction).

> Sitting at his office computer, Mr Smith prepares the files he needs for a business meeting later on that day and books a flight and a hotel room. Later he takes a random smart phone from the department's supply of mobile devices and logs on to the remote instant personalization server by means of a wireless network connection. As a result, his personal customizations, settings, and user data, which are maintained at the personalization server, are transferred onto the device. Mr Smith then synchronizes the device with his office computer, upon which the latest documents, calendar entries, contacts, tasks, and files are copied to the smart phone, including the booking reference code for his flight later in the evening, and the door-lock pin for his hotel room he had booked online. To fetch his luggage, he goes back to his apartment, where he also changes his suit. He then takes a cab to the airport, where he meets with a colleague who accompanies him on the trip. In the meantime, his smart phone wirelessly connects to the remote instant personalization server to perform a backup of new or changed personal data.
>
> Later, at the check-in counter, Mr Smith discovers that the flight is delayed. He also notices that he left his smart phone in his apartment. He therefore asks his colleague to borrow him his device for a few minutes. His colleague agrees and logs off his device, upon which the device loses all personal user information. Afterwards, Mr Smith logs on to the instant personalization service. From the list of his own personalized devices, he selects the smart phone he forgot at home and initiates a remote log-off. For personalizing his current smart phone, he only chooses to personalize his files and contacts, which are then retrieved from the server. Mr Smith now transmits his flight booking details to the clerk's computer via near field communication, and checks in for the flight. He then selects the phone number of his business partner on the smart phone, calls him to inform him of the delayed flight, and asks him to postpone the scheduled meeting. Afterwards he logs off the smart phone and returns it to his colleague. After the meeting took place, Mr Smith arrives at the hotel where he picks up an unpersonalized smart phone at the hotel lobby, which he can keep for the duration of his stay. He personalizes the device, proceeds to his room, looks up his door-lock pin now stored on his temporarily owned device, and unlocks his room. Before going to sleep, Mr Smith uses the smart phone to check his e-mail, to take some notes, and to call his wife. The next

morning Mr Smith leaves the hotel to return to the airport. He forgot his borrowed device on the table in his hotel room, but he is not worried about his personal data: the smart phone has automatically logged-off Mr Smith after one hour of inactivity, wiped all private user data from its memory, and is now ready to be used by other hotel guests.

7. Case Study: Dependable Computing Through Input/Output Diversification in the ETHOC System

In the following, we present our work on the ETHOC system (short for "**E**very**T**hing **H**as **O**nline **C**ontent"), which enables users to attach online information and functionality to physical objects. Concretely, the ETHOC system performs the creation, administration, and intermediation of online resources related to paper documents. To information providers, it offers a Web-based author portal for generating unique IDs that can be printed as barcodes and for associating online content and actions to printed material. To users it offers simple means of interacting with virtual counterparts of printed documents using a variety of devices, such as mobile phones or PDAs that support the Wireless Application Protocol (WAP) [Ope06], and it maintains a personal access history for each user.

The ETHOC system was developed as part of the "Entry Points" project at ETH Zurich, which dealt with applying ubiquitous computing ideas to a campus environment.[1] The ETHOC system allowed us to examine aspect of linking virtual and physical elements in such a campus environment, as it enables users to attach virtual counterparts to physical objects. In our case, for practical reasons and hardware availability, we focused on the use of printed material and barcodes as a linking technology. However, the underlying approach we chose can easily be adapted to accommodate any kind of physical object in arbitrary environments, and to integrate any linking technology for which there is suitable hardware available in the targeted user community.

With regard to dependability, ETHOC serves as a case study of applying *input/output diversification* as a design principle for interactive ubiquitous computing systems that have to be flexible, user-centered, and dependable. The ETHOC systems empowers the user to interchangeably employ a diverse set of user interface devices for collecting and accessing information attached to physical objects (here: printed documents) found in the surrounding smart ubiquitous computing environment (here: smart campus environment). Concretely, the user can choose among several different types of personal mobile devices including WAP-enabled mobile phones, personal digital assistants (PDAs), and notebook computers, as well as public web terminals or networked desktop computers.

[1] Entry Points was part of a larger initiative at ETH Zurich, named "ETH World" (see `http://www.ethworld.ch/`), whose goal it was to establish a virtual campus in addition to the existing physical campus. ETH World provided a virtual environment that sustained the university community, supported research, teaching, and learning, unified various university services, and was used by all people working or studying at ETH Zurich.

7.1. Providing Physical Hyperlinks into a Virtual Campus

The university campus is an interesting application environment for applying ideas from the field of ubiquitous computing [Wei91, Wei98]. In a campus environment, a substantial number of users share a large amount of their information needs. These needs include information about schedules, locations of class rooms, lectures, assignments, lab equipment, presentations, seminars, sports events, student ads, etc. Much of this information is directly related to objects, places, and people situated within the campus environment. The exhaustive installation of wireless computing facilities such as Wireless LAN, together with small handheld devices and technologies for detecting objects or locations, make it possible to satisfy the information needs of users in a campus environment in new ways.

While the World Wide Web with its manifold services in the domain of teaching and research exists mainly in the virtual world, a *ubiquitous computing campus infrastructure* is supposed to augment the physical campus infrastructure and to be closely related to the physical entities, places, and people within it. To achieve a close coupling between the physical and virtual campus, not only the parallel development of physical and virtual architecture are needed, but also their dense interweaving.

By embedding *physical hyperlinks* into the campus and attaching information to physical objects, visible entry points into the information space are created, enabling a natural interaction between the physical and the virtual environment and thus providing ubiquitous access to it. By linking the virtual to the real world, new interaction patterns emerge on both sides.

7.2. Entry Points into a Ubiquitous Computing Campus Environment

In [Fit93], Fitzmaurice discusses *situated information spaces* – real environments, in which electronic information is associated and collocated with physical objects that act as *information anchors* and provide a means of partitioning and organizing the information space. They provide "hot spots", which are roughly equivalent to what we call *entry points*, and serve as retrieval cues for users. The result is that information is situated and grounded in the physical context to enhance user orientation and ease of use.

A virtual campus as we envision it can be seen as an instance of a situated information space. For the usefulness and usability of a virtual campus it is vital to make it accessible from the physical objects and the physical environment of the people involved. To reach this goal, visible entry points are embedded into the physical campus environment that lead to items in the virtual campus infrastructure. The visibility of the entry points ensures that the virtual campus is present in the consciousness of its users and used in an everyday manner.

A way to realize this is to attach so-called *physical hyperlinks* to physical entities. Physical hyperlinks couple real objects to *virtual counterparts*, which represent physical objects in the virtual world and provide online resources and online functionality. Virtual counterparts also process events and capture relationships with

other virtual counterparts. Relationships between virtual counterparts are dynamic and evolve over time as a result of user actions and other events.

Potential relevant information related to physical objects might concern place and time as well as object-specific information or online information available on the Web [BR01]. Examples for such couplings in the context of a university are

- lecture halls that are associated with their occupancy status and reservation plan,
- technical equipment that is connected to its maintenance schedule or usage information,
- flyers, posters, and announcements that are linked to background information, electronic calendar entries or ticket reservation systems, and
- lecture handouts and exercise sheets that are coupled to related multimedia content or exemplary solutions.

Not only passive information, but also online functionality and actions can be triggered when following a link from the physical object to its virtual counterpart. This includes, for example, performing reservations, storing calendar entries on the user's mobile device, triggering e-mail messages, or signing up for event notifications.

To effectively support the everyday activities in research and teaching, the members of the university community have to be given the opportunity to actively participate. An easy-to-use system is needed that enables students, assistants, faculty, and staff

- to augment printed material they create as well as physical resources they use for teaching and research by online content and functionality, and
- to interact with the virtual resources using a variety of mobile and stationary devices of potentially limited capability.

These general ideas form the foundation and starting point of the ETHOC system.

7.3. Overview of the ETHOC System

After having introduced the concepts of entry points and virtual counterparts in the previous section, we now give a general description of the ETHOC system.

In its current form, ETHOC focuses on paper documents as instances of popular real-world objects of a university campus. The tasks that the ETHOC system performs are the creation, administration, and intermediation of online resources related to such documents. The system has two interfaces: one for information providers and another one for information users. The first interface is a Web-based author portal for generating and embedding ETHOC codes into documents. It also takes care of managing associated online content and document meta data, and of specifying the associated actions. The second interface offers simple means to interact with virtual counterparts using a variety of different devices. Examples

range from WAP-enabled mobile phones equipped with attached barcode readers, over PDAs with wireless connectivity, to the full fledged ETHOC browser for Java-enabled PCs and laptops.

Figure 7.1.: Picture of a printed ETHOC code with unique ID 0000000073 displayed both in human-readable form and in barcode representation, together with the former URL of the ETHOC Web portal at ETH Zurich

An *ETHOC code* is a unique identifier for linking a physical object (here: a printed document) with its virtual online representation. A printed ETHOC code shows the identifier in barcode representation and in human-readable form, and the URL of the former ETHOC Web portal at ETH Zurich (`http://ethoc.ethz.ch`, now offline). A picture of a typical ETHOC code is shown in Fig. 7.1.

The motivating scenario described in Sect. 6.3.4 explains how the ETHOC system is used from the author's as well as from the client's perspective.

This scenario vividly illustrates the notion of *environment-mediated communication* [Gel98], where electronic information is dispersed throughout the environment, enabling casual interaction and anonymous communication. Environment-mediated communication describes how communication between persons is mediated by instances of the physical environment. It is motivated by the traditional use of the physical environment for the mediation of information between people, in which messages are left at specific places or particular objects for later retrieval by other people: ads and announcements are posted to bulletin boards, post-its are pasted on office doors or folders. Information dispersed in this way is bound to a physical object or location and therefore reduces information overload. Information is organized according to physical entities and information of only local relevance is filtered by location.

The short interaction time of just scanning an item that is then automatically inserted into a personal online history is crucial for usability in a mobile environment. The action does not require much effort and takes just a few seconds. The user can later review the scanned items and is not distracted from the current activity.

The scenario also shows that a variety of devices can be used to interact with the virtual counterparts. Information that is situated in a real world context can be picked up in the originating context using an unobtrusive mobile device. The information can later be reviewed and interacted with in a more suitable situation, using a stationary device with better display capabilities. That way, mobile and stationary devices complement each other. Mobile devices have severe size restrictions and limited interaction capabilities, but are easy to use in a mobile context of an everyday real-world situation. Stationary devices are used outside of the situational context, but offer richer user interface capabilities.

The majority of the information picked up might be immediately useful to the user in the current situation, while some other information might be more useful at a later point in time, potentially in another context. ETHOC supports this by

automatically storing scanned ETHOC codes in the user's personal online history of scanned objects.

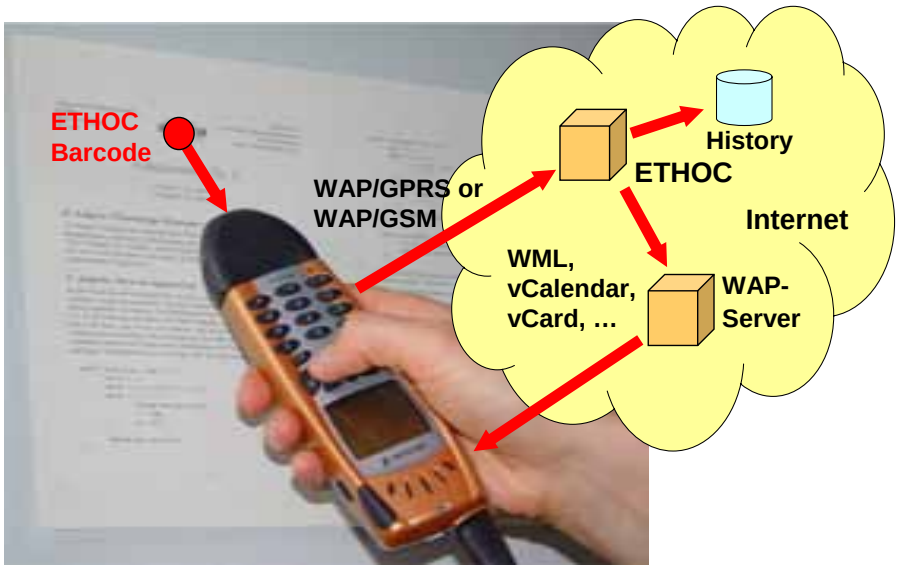

Figure 7.2.: Scanning an ETHOC code using a mobile phone with an attached barcode reader

Figure 7.2 shows what happens when the user scans a barcode with a mobile phone. The wireless communication technology used in the displayed case is WAP over GSM/GPRS [GSM06], but it could also be Wireless LAN [WFA06] or Bluetooth [BS06] instead. The scanned ETHOC ID is sent as part of an HTTP request to the ETHOC server. In addition to the ID of the scanned object, the request contains information about the client device capabilities, which is used to render the result in a format the client can display.

The ETHOC server stores the scanned ID in the user's personal online history, where it is available for later retrieval using other devices. Depending on the device capabilities, an HTML or WML page that contains hyperlinks to the document's online content is generated and sent back to the client device. By following the links, the associated online content – such as background information, contact information, or calendar entries – can be retrieved. Which functionality of the virtual counterpart is available at a time depends on the device currently in use; the automatic insertion of a calendar entry, for instance, may only make sense for a mobile phone or PDA, but not for the Web browser used in a public Internet café.

Even though the display size of current mobile phones is often rather limited, mobile phones in general represent useful tools in the showcase described earlier. Using the standard vCalendar [DS98] format, calendar entries can automatically be inserted into the user's personal calendar to act as reminders for deadlines or events. Using the vCard [DH98] format, phone calls can immediately be placed. Last but not least, WML has capabilities that come close to those of HTML. Fig. 7.3 shows a document as displayed on a mobile phone. The left screenshot shows the ETHOC ID and the title of the document, the middle and right parts show information about the document author. For example, based on the functionality provided by the WML page, the user can opt to store the author's address in the calendar of the phone.

Figure 7.3.: On the mobile phone, a WML page is shown as the result of scanning a paper document

7.4. ETHOC System Architecture

In the ETHOC system, printed documents are enhanced with online resources and functions by means of so-called virtual counterparts. Currently, barcodes are used to provide a unique ID that unambiguously links the physical and virtual document representation. (see Fig. 7.4).

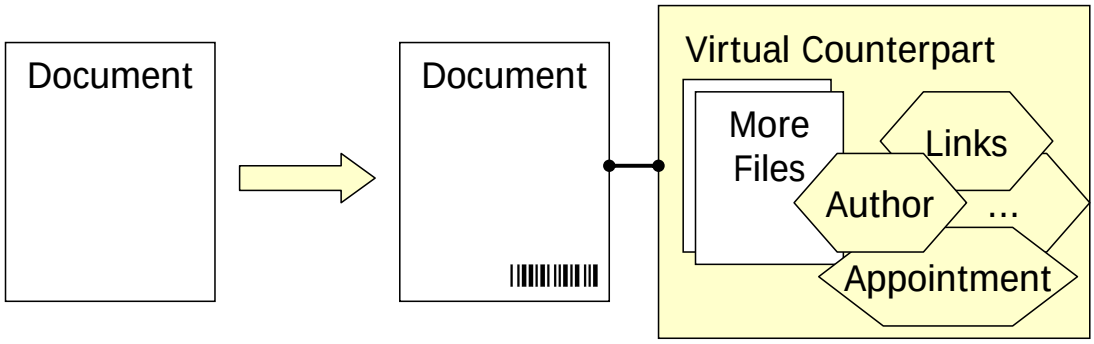

Figure 7.4.: Enhancement of a physical document with online resources by means of a virtual counterpart. The barcode serves as a unique object identifier, which unambiguously links the physical document with its virtual representation

The augmentation of printed documents requires authoring support for authors of documents. Besides, as we have seen earlier, users should be in a position to scan and view documents using different available everyday life technologies, such as a notebook computer, a PDA, or a mobile phone. This helps to maximize both the ease of use of the system and the availability of the information and services it provides. Further, a common data format is required for allowing a flexible and efficient processing. Finally, the system has to persistently store virtual documents and manage their life cycle and functionality, including the management of barcode IDs or of a user's document history.

The ETHOC system was designed to accommodate these requirements. It features a modular architecture which is flexible and extensible. In the following, we give an overview of the system architecture, describing the main system components that can be categorized as XML-based data processing, authoring support, client support, and back-end data management.

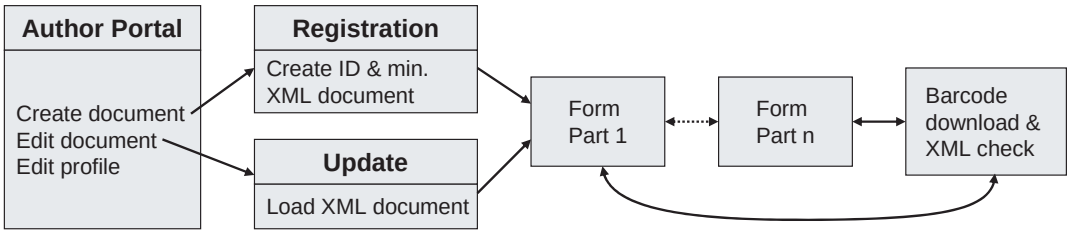

Figure 7.5.: Flow of operation for the registration of new documents and for editing and updating existing documents in the author portal

XML-Based Data Processing. The generic description language XML was chosen to model the data structures of virtual counterparts of arbitrary objects in the ETHOC system. Extensible Stylesheet Language Transformations (XSLT)[2] are used to convert internal XML data structures into client-readable data representations. For each object type different XSLT descriptions are stored in the ETHOC database (e.g., for HTML, WML, or XML data representation), together with suitable mappings for various client devices. Then, whenever a document is requested, the type of the client (or user agent) is evaluated and the data transformed corresponding to the best matching XSLT description.

XML is also used to store and maintain the configuration of system components in a flexible manner. The number and type of modules available in the authoring portal, for example, are stored in an XML document that can be edited to add new tools or modules. As a side effect, the broad range of advanced tools for parsing, processing and transforming XML (see [Xal02, Xer02], for instance) contribute to the versatility of XML-based data processing. With respect to virtually augmented documents, the XML structure of a virtual document counterpart consists of four main sections: author information, document related information, contact information, and actions.

Authoring Support. The authoring portal[3] provides authors with tools to create and modify documents and to edit the personal profile. In addition, the authoring support also includes modules for author registration and authentication.

After successful log-on, an author may either create a new or edit an existing virtual document counterpart. In the first case, a new minimal XML document is generated, and the user is guided through different forms to fill in required and optional information. In the second case, the existing XML document is loaded and displayed. In both cases, the editing process operates directly on the XML representation of the document counterpart. With respect to its structure, the XML document is divided into different sections. For each document section there exists a distinguished module, represented by a separate servlet application which is in charge of displaying and editing the current document fragment, and of the transformation and validation of the data entered by the author. An author may either follow a preconfigured sequence of authoring tools to complete a new counterpart, or alternatively just select the modules of interest. Once the author has entered all obligatory information, he or she can download an ETHOC ID (barcode) and

[2]See `http://www.w3.org/Style/XSL/`
[3]See `http://ethoc.ethz.ch/`

insert it into the electronic version of the document that is to be printed. Before a new document counterpart is activated, a summary of the added meta data and selected functions is presented to the author by a special finalizing servlet.

The possible operation sequences for the creation or modification of virtual document counterparts in the author portal are illustrated in Fig. 7.5. The document counterparts are edited in different form parts. The functionality of each stage in the diagram, including the various form parts, is performed by an independent module. New modules can be introduced by updating the global configuration (XML file) of the author portal.

Currently, the following authoring tools (modules) have been implemented for virtual document counterparts: *author notification* for life cycle management support, a *feedback questionnaire*, and a *news client* to read or write news articles related to the current document.

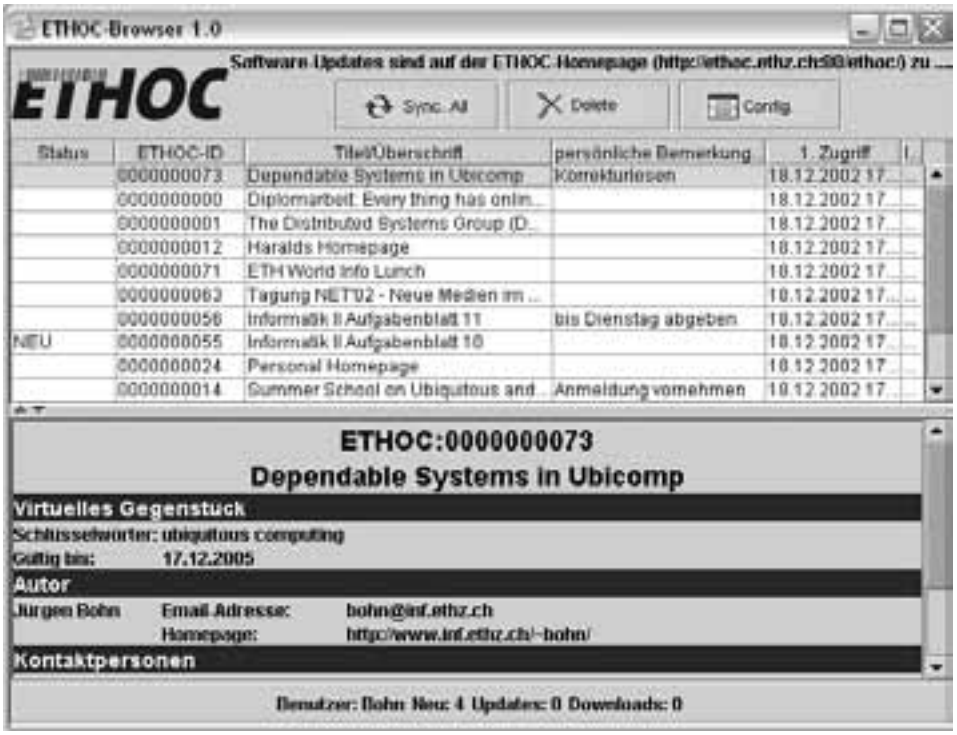

Figure 7.6.: Screenshot of the stand-alone ETHOC Browser

Client Support. The ETHOC system supports three different ways of accessing the virtual counterparts of physical documents. First, it provides a Web-based user interface that can be used on any device with an installed standard Web browser. Second, for small, resource-limited mobile devices such as mobile phones, a WML interface has been implemented. Since small devices often have limited capabilities (such as a lack of viewers for PDF and PostScript documents), and due to the restrictions imposed by WML, the WML portal only grants access to a limited number of features and items of information of a document counterpart. Third, for devices that are capable of running Java (e.g., laptops or PDAs) we provided a designated application, the ETHOC Browser (see screenshot in Fig. 7.6), which provides advanced features such as document caching and automatic synchronization whenever document updates occur on the ETHOC back-end server.

If the client configuration and the available hardware permits, a barcode on a document can be scanned automatically by means of a mobile barcode reader. Otherwise, each client offers a user interface for manual input of the corresponding numeric barcode value that is displayed on the corresponding ETHOC code.

Back-End Data Management. An important task of the ETHOC system is the management and persistent storage of virtual document counterparts that consist of the XML descriptions and the attached objects (files, movies, etc.). The back-end database system – we use MySQL for that purpose – keeps track of all documents that are viewed by a person, maintaining a personal history for each user. Further responsibilities of the back-end data management include the accounting of assigned ETHOC codes and the administration of user profiles.

7.5. Results

The ETHOC system provides *physical entry points into a virtual infrastructure* by linking physical documents to virtual counterparts. Further, its modular design contributes to an increased *flexibility* and *extensibility* of the system.

By supporting a variety of heterogeneous clients and ubiquitous computing devices, the system allows accessing virtual object counterparts in various different situations and locations, such as in an Internet café using a Web browser, on the train using a mobile phone, or at a communication hot spot at public places (e.g., at the airport or train station) using a PDA with wireless LAN or Bluetooth connectivity. Because of fully interchangeable clients and with the assistance of the user's personal access history, any other client is suitable to retrieve the information about previously scanned documents in case a previously used client device is left behind unintentionally or gets lost. This user-friendly behavior increases the *ease of use* of the system.

At the same time, these characteristics also positively influence the reliability of the system: By allowing the interchanging use of multiple, diverse clients, a *diversification* of the information access is achieved: individual devices may be lost, left behind, or broken, but the information that has been retrieved *once* remains accessible through alternative means input/output. While a user may own just one device of a kind (e.g., one mobile phone, one PDA, or one laptop), the functionalities of all his devices overlap to some degree when taken together (functional redundancy, see Sect. 4.5.7). As a consequence, the various devices that possess some means of wireless communication and web content displaying and editing capabilities can be used independently from each other to connect to the ETHOC service, reducing the dependence on a single device or technology. Thus the functional redundancy of the available heterogeneous devices is exploited, yielding a higher *accessibility* and *availability* of information and services on the client's side.

Furthermore, the ETHOC system is capable of tolerating transient disconnectivity, as it explicitly supports *disconnected operation* of all mobile client devices that possess a standard Java Virtual Machine to run the stand-alone ETHOC browser application. The ETHOC browser performs local caching of on-line resources for off-line operation, including the automatic synchronization of documents updated on the server or newly scanned in with a barcode reader.

For identifying printed documents, we used one-dimensional barcodes (see also Appendix B.3.4). For identifying other types of objects, alternative identification technologies could be more suitable, such as using active radio beacons to tag rooms or other locations. In this case, however, the chosen identification technology either has to be ubiquitous and pervasive in the sense that it is available in a sufficiently high proportion of all user interface devices of the targeted user community, or the system has to provide alternative means of performing the identification without the need of special hardware. In our case, the ETHOC codes themselves feature functional redundancy as described in Sect. 4.5.7, as the identification can be performed either by scanning the barcode or by manually typing the corresponding human-readable identification number (see Figures 7.1 and 7.2, respectively).

7.6. Experimental Evaluation

The use of barcodes as the underlying identification and linking technology proved to be very practical. Barcodes can be printed on paper and do not require special (pre)processing, as is the case with passive radio frequency identification (RFID) tags which need to be initialized and manually attached to physical documents.

Also, instead of requiring the author to manually place an ETHOC barcode inside the documents, a printer driver that combines the authoring procedure with a fully automated barcode integration would further improve the ease of use and robustness of the authoring procedure (e.g., preventing the user from attaching an ETHOC code to the wrong printed document).

Concerning practical experiments, we annotated exercise sheets with ETHOC codes in an undergraduate lecture. This allowed us to timely provide source code fragments and exemplary solutions to students. By means of the integrated news client module, we successfully setup custom news groups for the discussion of specific exercises and coupled them with the corresponding exercise handouts. Apart from these promising initial experiments, the system has not yet been applied on a day to day basis with larger user groups.

In our group we further experienced that the perceived convenience, usefulness and fun factor increased noticeably with the availability of barcode scanner devices for automatic barcode scanning, such as a mobile barcode reader attached to a mobile phone, or a small and handy stand-alone wireless barcode scanner. In this respect the extra costs for the required add-ons still constitute a handicap, especially since popular consumer products such as PDAs and mobile phones are not equipped with barcode scanners per default. However, as each ETHOC code also contained a human-readable version of its respective identifier, the possibility of manually "scanning" barcodes enabled students and other users to participate in the ETHOC experiments even without the availability of barcode scanner hardware. Further, the built-in cameras of advanced cellular phones can also be employed for barcode detection, removing the need of a dedicated barcode reader [Qod06].

The ETHOC system therefore can be set up as a truly open system that does not pose access restrictions based on the need of special hardware, which is in accordance of the open-world assumption of ubiquitous computing systems described earlier in Sect. 3.3.9.

As part of future work, field studies with non-expert users, such as with students from fields other than computer science, should be carried out to assess the usability and user acceptance of the ETHOC system.

7.7. Conclusion

While in the current version of the ETHOC system we mainly dealt with document-type objects and their counterparts, the overall architecture has been designed with the intention to support arbitrary real-world objects. In future versions of ETHOC, we plan to extend the range of supported object types and authoring modules. We also intend to investigate concepts of context-aware retrieval of online content, such that the provided information that a user receives not only depends on the ID of an object, but also on context like the identity or location of the person.

Our main research goals were to get insight into the design of virtual counterparts that augment physical things in the real world, and to investigate novel means interaction with smart ubiquitous computing environments. Further, by applying of input/output diversification, the accessibility of interaction devices could be maximized for the individual user, resulting in a high degree of flexibility, ease of use, and availability of information and services provided by the ETHOC system. For direct interaction with online content and functionality of augmented physical objects found at the current location, a person can interchangeably use (a) his or her favorite personal mobile devices while on the way and (b) any available public terminal or other user device (e.g., personal user devices temporarily lent from other persons) with Web browser capabilities. For location-independent a posteriori access to arbitrary services and data of smart objects, a user can browse through his personal access history maintained on the ETHOC server, or manually enter arbitrary ETHOC codes. By using ETHOC clients supporting local caching, a user can also access ETHOC resources during disconnected operation.

7.8. Related Work

7.8.1. Linking Physical with Virtual Spaces

At Xerox PARC the *ParcTab* ubiquitous computing environment [WSA+97] was established to augment everyday artifacts with computational capabilities. Want et al. [WFGH99] describe various techniques that were used in that system to couple the physical and the virtual world. Paper documents, books, and other artifacts were equipped with RFID tags and related to corresponding Web pages and services. In ETHOC we use barcodes, because they are easier to handle and are well suited to being printed together with the paper document.

HP's *cooltown* project[4] is a pragmatic approach for the realization of "smart" environments, in which physical and virtual worlds move closer together. The goal of the *cooltown* project is to establish a "real" world wide web, in which users can pick up URLs that are embedded in the real world and in which people, places and things have an immediate presence [Tim02]. Unlike in ETHOC, the virtual counterparts are represented by ordinary Web pages in this model.

In the *Sentient Computing* project[5] [ACH+01] of AT&T Laboratories so-called "smart posters" were developed. Smart posters are printed and attached to a wall and contain sensitive areas that are linked to specific actions. A small hand-held

[4] See `http://www.cooltown.com/`

[5] See `http://www.uk.research.att.com/spirit/`

device called "bat" is used to select individual sensitive areas. The *Sentient Computing* system is based on an ultrasound location system and is rather expensive to install. In contrast to that we focused on cheap and immediately deployable technologies that require less implementation effort and costs.

The *WebStickers* system [LRH00] uses barcode stickers to access and organize bookmarks on the Web. The stickers are attached to everyday objects that act as physical representations and reminders of the associated Web pages. The bookmarks are therefore organized according to the context of the physical workspace.

To our knowledge, apart from *WebStickers* which does so in a limited way, none of the systems mentioned above provides an authoring interface that allows a group of people to actively engage in the content generation and publication process. However, we consider it as crucial that the people interacting with the system are also in a position to actively design and shape their information environment. As described earlier, ETHOC provides a Web-based author portal for that purpose.

7.8.2. Input/Output Diversification

There has been a considerable amount of work on providing device-independent ubiquitous data access from heterogeneous devices based on a file-system-oriented approach. UbiData [ZHH03] probably is the most comprehensive and representative system in this area. It constitutes a further development of the mobile file system idea that was addressed by the early CODA [KS92] file system [HHZK01]. The three main goals of UbiData are (1) any-time, any-where access to user data even in the face of temporary disconnectivity or weak connectivity through low bandwidth and high latency networks, (2) device-independent data access enabling the user to switch among devices with different capabilities and maintain data access, and (3) application-independent access to data to allow users to modify portions of documents and files belonging to classes of related applications [HH04]. The main mechanisms applied in UbiData for achieving these goals are automatic data selection, hoarding, and synchronization [HKZ02], and transcoding, which together enable continuous availability of data regardless of user mobility and disconnection, and regardless of the mobile device and its data viewing/processing applications. In the process, maintaining currency (up-to-dateness) and consistency of data on mobile devices, support for mobile transactions [LH02], and meta data and profile management (for building so-called converged networks in which users can freely use a host of services from cooperating providers through a single sign-on [SHLX03]) are considered the central challenges.

Obviously, by supporting different clients, UbiData and related systems implicitly provide support for input/output diversification with regard to *data* access. In contrast to the *data-centric* view and file-system oriented approach of UbiData where a user's data is shared and processed by a number of interchangeable, highly distributed and possibly mobile entities, we pursued an *interaction-based* approach in the ETHOC system. In ETHOC, hoarding and transcoding of data is not an issue, and data synchronization and representation of minor importance. Instead, the main objective is to enable the user to enhance physical objects (here: printed documents) with online information and *functionality*, and to empower people to *interact* with such augmented objects and – through these objects – with other people. In this context, input/output diversification serves as a method to provide

the user with redundant means of interaction to increase the flexibility, ease of use, and accessibility of online services and information linked to physical objects.

Acknowledgments

This chapter is based on joint work with Michael Rohs [RB03]. The author further wishes to thank Nikolaos Kaintantzis who was instrumental in implementing the ETHOC system [Kai01], as well as Harald Vogt for implementing the news client module.

8. Device-Independent Interaction by Means of Instant Personalization and Temporary Ownership of Handheld Devices

As we increasingly depend on inexpensive handheld devices at work and in daily living, ensuring the accessibility of those devices and the availability of the personalized services they provide becomes a major challenge. Further, mobile devices are currently still poorly integrated, as existing mobile computing infrastructures often lack support for automatic synchronization and data management across the various devices owned by a user [MS03].

In the following sections, we present a system for the *instant personalization and temporary ownership of mobile devices* that addresses these issues. The system enables the user to make the transition from requiring a specific *individual* device to utilizing *any* device at hand: instant personalization empowers the user to instantly turn arbitrary devices into a fully personalized device containing both the person's user data and meta data. This significantly raises the degree of redundancy of devices accessible to the user from one to a potentially unlimited number of devices of a certain type, enabling the user to tolerate hardware and inaccessibility faults in particular as described earlier in Section 4.1. As a result, the accessibility of specialized functionality offered by personalized handheld devices and the availability of personal user data is increased. The system that we prototypically implemented further provides support for (1) periodic data backup, (2) data recovery, enabling the user to retrieve private data from physically unavailable devices, and (3) data confidentiality protection, assisting the user in preventing illegitimate data access on behalf of third parties in case a personalized device was lost, stolen, or unintentionally left behind.

8.1. Instant Personalization of Mobile Devices

In this section, we first discuss the roles that user data and meta data play in personalization. Then we define the terms of "instant personalization" and "temporary ownership", which are central to this chapter. Finally, we show how the instant personalization of handheld devices is experienced from the user's perspective.

8.1.1. The Roles of User Data and Meta Data

Today, users of handheld devices typically personally own one device of a kind, each of which serves a particular purpose and therefore provides functionality for which it has been designed and optimized. A mobile phone is optimized for making phone calls, a PDA is convenient for keeping track of appointments or for taking quick notes, or a smart electronic book (e-book) is a compact means of carrying with you the content of multiple books while enabling the user to search for words and phrases, to append written annotations, or to add bookmarks, for example. A handheld device may offer several of such services if it meets the requirements with respect to technological capabilities and ease of use (e.g., a PDA with a high-quality display can be used as an e-book, or a smart phone combines the capabilities of PDAs and traditional mobile phones).

The operation of handheld devices involves *personal user data*, either because the primary purpose of the device is to edit and manage such user data (e.g., taking notes, updating an existing contact, or adding a new appointment), or because the data is needed for the provisioning of specific services (e.g., a reminder service needs access to the user's personal calendar, a smart phone utilizes the user's list of contacts to retrieve the phone number of the person that is to be called, an e-book requires the digital version of the book the user wants to read). The *individuality* of a handheld device, however, is mainly determined by the user's individual preferences and device settings which make up the so-called *meta data*. Such meta data not only improves the efficiency and the ease of use of certain services provided by the mobile device (e.g., bookmarked web addresses, application-specific defined shortcuts, customized views and program settings, etc.), but often constitutes an essential element of these services (e.g., mail account settings, remote file server settings, passwords, etc.).

Besides user data and meta data, the personalization of mobile devices may also include the temporary installation of *personal applications* which are not part of the standard software that comes with a specific type or brand of device. In principle, if an application is self-contained, it can be treated as ordinary user data: it simply has to be copied into the correct folder on the mobile device. The deletion of such personal software later on is straightforward, too, since it is sufficient to delete the previously copied files. Otherwise, apart from legal issues, it may not be advisable to install applications that are not self-contained, since they often require the presence of certain libraries or runtime environments, or because their installation procedure may not be fully reversible or cannot be performed in an unattended fashion, for example.

8.1.2. Instant Personalization and Temporary Ownership

The goal of an *instant personalization* of mobile devices is to transform an arbitrary device, devoid of any personal user information, very quickly into a fully personalized device and with minimal involvement of the user. Further, the instant personalization process should be started the very moment the user actively and deliberately initiates it, no matter when (*anytime*) or where (*anywhere*).

So with an infrastructure for instant personalization in place, it becomes possible for a user to take *temporary ownership* of arbitrary devices he or she does not own personally but which are only available to him or her for a limited period of time.

Once the instant personalization of a device is completed, it is indistinguishable from a personally owned device of the same type with respect to the particular personalized functionality and user data.

To be in a position to instantly personalize an arbitrary device on demand, anywhere and anytime, the device requires access to a background service (i.e., a service offered by the background infrastructure) that provides the user's personal data and meta data. The access should favorably be performed by means of a wireless connection in order to not impede device mobility and portability. Further, the mobile device itself needs to know how to retrieve and install the data needed for instant personalization. Once the device is no longer needed, it has to be able to write back those parts of the data which have been modified since the personalization was effected, before being "released" from service and becoming available again to be temporarily claimed and personalized by future users. So the *release* of a device is the inverse operation to the personalization procedure.

In situations where personal data on a mobile device is utilized in a non-manipulative fashion (e.g., personalizing a mobile phone just for making a few quick phone calls), it may be practical to perform a *read-only* personalization where any personal user data can simply be deleted when the device is released at a later point in time. By analogy, if we wish to explicitly state that any modified personal data has to be copied back to the server when the mobile device is released, we speak of a *read-write* personalization. Note, however, that even if personal data has not been deliberately modified by a user, potentially useful meta-data such as the user's call history or e-mail history is not preserved in the case of a read-only personalization.

8.1.3. User Experience

From the user's perspective, the process of instant personalization of mobile devices is experienced as follows:

A user picks up an arbitrary *blank* device (i.e., a device devoid of personal user information) that is physically available in the current place at the given time. The user takes temporary ownership of the handheld and logs on, using a personal user name or fingerprint for identification, and a password for authentication. Hereupon the device automatically downloads the user's device-specific individual preferences and settings, the meta data, together with his or her device-dependent user data from a dedicated server in the background infrastructure, using a wireless connection. Thus the user's preferred personal configuration is re-established on the device. Now the user is able to use the device (e.g., the personalized phone, PDA, or e-book) as if it were exclusively owned by him or her, exploiting the capabilities of the device to its fullest, with the personal user data (contacts, appointments, notes, etc.) as well as the personal meta data (e.g., mail server settings, passwords, bookmarks, shortcuts, customized views and program settings, self-contained applications and tools, etc.) installed. When the device is no longer needed, the user simply logs off, upon which the latest modifications of device settings or user data are written back to the back-end server. Finally, all the user's personal user and meta data are wiped off the device. This restores the original uninitialized blank state of the device so that it becomes available again to other users.

8.2. Design Goals

By means of instant personalization and temporary ownership, we realized the following conceptual design goals:

1. Higher *availability* of personal user data by providing anytime, anywhere access;
2. *interchangeability* of handheld devices to increase the *accessibility* of personalized device functionality;
3. support for *disconnected operation*;
4. *periodic data backup*;
5. *recovery* of personal user data and
6. *protection of confidentiality* of personal user data stored on personalized devices that are physically unavailable;
7. support for user-friendly *life cycle management*.

With the provisioning of instant personalization of mobile user devices as a service, the first two goals were implicitly realized: during personalization, user data is copied onto the device, and the personal meta data is automatically installed, which yields the personalized device functionality. Note that device interchangeability and device-independence have different meanings. Device *interchangeability* refers to the fact that I can easily change my device by virtually "moving" my personal user data and meta data from one device to another device of a certain type. In doing so, the device-specific characteristics in terms of usability and functionality are retained. The aim of *device independence*, however, is typically regarded as to provide an abstract, device-independent means of performing a task. In this case, the original qualities of the particular devices are not preserved or considered of secondary importance only. Examples hereof are the virtual network client or the remote desktop access described in Section 8.6.

The instant personalization of mobile devices anywhere and anytime requires connectivity to the instant personalization server as part of the background infrastructure. However, once the personalization is completed, the personalized mobile device no longer needs to stay connected but can operate in disconnected mode, thus supporting *disconnected operation*. In case of a read-write personalization, connectivity is again required when the device is to be released and data has to be sent back to the instant personalization server.

Portability and usability owing to a comparably small form factor are some of the key advantages of handheld devices. However, as the number of small personal devices a user relies on grows, it becomes gradually more likely that a certain device is physically unavailable when needed. This may be because the user simply left the device behind (e.g., the user forgot to take the personal mobile phone out of the jacket worn the day before). A mobile device may even become permanently unavailable in case the user loses it someplace, or if it gets stolen. Apart from the material loss, this led us to two further concerns. One was the *recovery* of personal user data stored on the device, especially if no recent backup of the data exists.

Another concern was the *protection of data confidentiality*, as the personalized handheld device may carry private information that should not be revealed to others, or confidential data such as passwords or credit card numbers.

Last but not least, the increasing availability of mass-produced and cheap personalizable mobile user devices and appliances leads to ever shorter product life cycles. In addition, small, portable devices are more vulnerable to physical damage and loss than more robust, conventional (stationary) computer devices of larger size. As a consequence, as people increasingly use and depend on mobile user devices, the rate at which a user has to replace his or her own personalized mobile devices increases accordingly. The product life cycles of mobile phones, for instance, have decreased to 12 months or less. At the same time, the majority of users expects immediate readiness of new devices (out-of-the-box experience) [GSI05]. As many acquired personal data and device customizations are likely to outlive the life cycle of the physical devices (such as contact lists and calendar entries, passwords, e-mail client settings, and so on), *life cycle management support* is important. Therefore a further goal of our system was to provide easy-to-use support for the rapid migration of personal user data and meta data in order to enable the instant readiness of replaced personal devices.

In our concept, these issues are addressed in the following way. First, a personalized device may perform an automatic release (auto-release) after a prolonged period of inactivity, triggered by a user-definable timeout. Second, the release of a personalized device can also be initiated by an external device, such as the remote personalization server or another device personalized by the user. In either case the client device first reconnects to the server to write back recently modified or added user data. As a result, the data of temporarily or permanently unavailable mobile devices is recovered (given that network connectivity is available and the batteries in the device are not depleted). Data confidentiality is preserved by erasing the user's data on the handheld device when the latter is released, which prevents the fraudulent use of private data. Additionally, if data confidentiality is paramount, the mobile user device can be configured to lock itself automatically after a short period of inactivity, preventing other users from accessing private information before the device completed the release operation.

Finally, user data is implicitly protected by means of data backups whenever a personalized handheld device is released and the modified user data is retransmitted to the server. However, a personalized user device may be continuously used for a longer period of time. If this device breaks down beyond repair during operation, all data that has been modified since the personalization is lost, too. To prevent this, the client can be configured to regularly reconnect to the instant personalization server in order to transmit recent changes in user data and meta data, thus realizing a *periodic data backup*.

8.2.1. Exclusive and Concurrent Personalization

Instant personalization benefits from the observation that a user typically only utilizes one personalized device of a kind, such as one mobile phone, one PDA, or one laptop, for example. In general, it is therefore sufficient to have only one device of a kind personalized in read-write mode at a time, removing the need for complex data synchronization and conflict resolution schemes. Consequently, in our concept,

read-write personalization currently is performed as an exclusive operation. This means that a device that has previously been personalized in read-write mode has to be released before the server allows the user to personalize another device of the same type in read-write mode. Note that the user may concurrently perform as many read-only personalizations on separate devices as desired, as they do not require further assistance on behalf of the instant personalization server. However, read-write is probably the preferable personalization mode since it also preserves additional meta data (such as a call history, for instance) that has been accumulated during the utilization of a personalized device.

It may happen that a user wants to instantly personalize a device at hand even though a previously personalized and currently unavailable device has not yet been released before. Then, instead of waiting for the automatic timeout-triggered release to occur, the instant personalization server can enforce the release of the unavailable device by means of a remote release-request sent via the network. If the remote device is not reachable, the user may choose to perform a read-only personalization for the time being and wait for the remote device to write back its modified data in the meantime. Alternatively, the user can override the read-write personalization lock on the server and enforce a new read-write personalization, upon which the server considers the data on the unreachable device as stale and no longer valid. From the perspective of a device which has been personalized in read-write mode but which is unable to connect to the instant personalization server, there are also two options: the device may either postpone the auto-release for a specified amount of time and try to reconnect in the meantime, or perform the release operation anyway, rating the protection of data confidentiality higher than data recovery.

Of course, computing power of the mobile devices and communication bandwidth allowing, data synchronization techniques as discussed by Zhang et al. [ZHH03] may also be used to support multiple concurrent read-write personalizations per user and type of device.

8.2.2. Cross-Platform Personalization

The instant personalization of a mobile device not only includes the transfer of personal user data to the device, but also the installation of the meta-data that is required for the smooth functioning of the particular applications the user expects to work with. However, user and meta data for a device often depend on the specific type of device, applying to concrete versions operating systems (such as Symbian OS for mobile phones, Windows CE or Palm OS for PDAs, for instance) and the standard applications associated with these operating systems. As a consequence, to widen the applicability of personal data, it is necessary to provide abstractions or generalizations for *device-specific* knowledge on how to automatically install or extract a user's data and meta information.

One solution are *standardized interfaces* to applications that are commonly integrated with certain operating systems or types of devices. Such a standardized interface already exists for the Microsoft Windows CE operating system (version 3.0 and later): the Pocket Outlook Object Model (POOM). It provides a generic API for manipulating contact, calendar, and tasks data. As these data make up an integral part of the personal user data and meta data typically used on a PDA, the

POOM interface contributes towards realizing a unified personal data management for Windows CE based handheld devices, irrespective of hardware configuration and manufacturer.

Another possible solution is the definition of device-independent *personalization profiles*. Such profiles could describe the structure and vocabulary of personal user settings and preferences. Once personalization profiles for mobile user devices have been defined and standardized, they provide an abstract and universal interface for manipulating personal user data and meta data across different hardware platforms and operating systems. An example for such a standardization effort are the Composite Capability/Preference Profiles (CC/PP) [Dev04] proposed by the W3C Device Independence Working Group, which have been designed to enable "access to a unified web from any device in any context by anyone". A similar goal is pursued by the development of SyncML. SyncML is intended as a single common data synchronization protocol that aims to deliver an open, industry-wide specification for the universal synchronization of remote data and personal information across multiple networks, platforms, and devices.

Note that even if device-independence is achieved at the software level, problems can still arise from the heterogeneity of manufacturer-dependent hardware components. Although supporting a similar operating system, mobile phones from different vendors may still significantly differ in terms of hardware control elements (such as different button layouts), for example. Thus the perceived ease of use of a successful instant device personalization may be negatively affected, especially if a user has difficulties adjusting to unfamiliar operating controls.

An alternative approach could be to upload a complete virtual machine image instead of performing a fine-grained personalization on the data element level. Such a procedure would implicitly retain all software installations and system modifications performed on the personalized device. However, this scheme has several drawbacks. Firstly, it would typically be necessary to transfer the complete (binary) image even if only a small portion of the user's personal data had been altered, thus increasing the data transfer load and impeding a customized personalization procedure. Secondly, it would no longer be possible to synchronize and combine data modifications from multiple devices since either all changes or no changes could be retained per image. Thirdly, the upload of complete images (including systems software and applications) onto arbitrary devices would presumably conflict with existing licensing policies.

While we are planning to eventually employ a generic personalization profile such as CC/PP or SyncML, our initial prototypes have been developed using a combination of the standardized POOM API, together with a set of operating-system-specific methods not supported by the API (e.g., manipulating the registry under Windows CE).

8.2.3. System Architecture

Our prototypical system for instant personalization consists of a client component (Instant Personalization Client), which is executed on the mobile devices, and a server component (Instant Personalization Server), which resides in the background infrastructure (see Figure 8.1). In the following, we describe the two components in more detail.

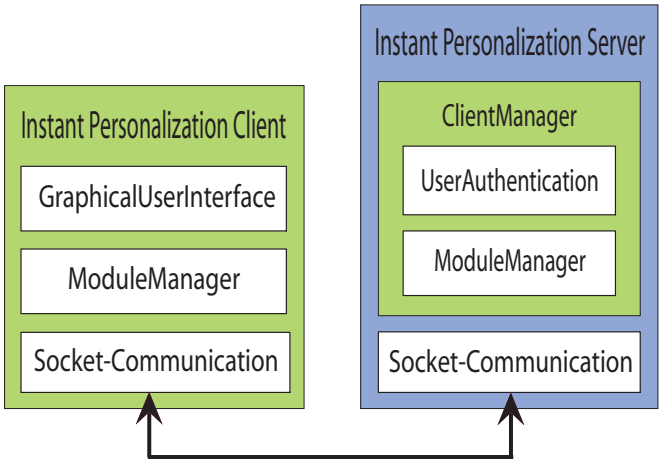

Figure 8.1.: Architecture of the Instant Personalization System

The *Instant Personalization Client* (IPC) is executed as a *persistent system process* on the mobile device. It is automatically launched whenever the device is started (e.g., after a reboot or reset). The IPC also features a *graphical user interface* (GUI) (see Figure 8.2 for screenshots of the GUI we implemented for our IPC client prototype). If the device is in the unpersonalized state, the GUI allows the user to log-on to the personalization server (see login screen dialogue), to choose the program modules that have to be personalized (see selection of personalization modules dialogue), and to specify the timeout (in minutes) for the auto-release function. Afterwards, the *module manager* performs the instant personalization for all selected personalization modules. After the device has been personalized, the user can choose the "release now" menu option to actively release it once the device is no longer needed. The user can also wait until the IPC-internal timer starts the auto-release operation (see auto-release notification dialogue). The IPC communicates with the IPS via *sockets* using TCP/IP connections. For the server-initiated release, a separate listener thread on the IPC listens to server requests.

The *Instant Personalization Server* (IPS) acts as a background service residing in the network. It manages the database which contains the users' specific data needed for the instant personalization of mobile devices. The IPS uses sockets with a fixed port number to listen to IPC requests. Whenever a user takes ownership of a blank device and initiates the instant personalization procedure, the IPC on that device connects to the IPS via a secure channel. The IPS *client manager* identifies the user and authenticates the corresponding password. On successful authentication, the module manager on the user's handheld requests personalization information for the desired personalization modules which are then returned by the counterpart module manager on the IPS. Otherwise, the connection is closed by the IPS. Similarly, if the user's personalized device is to be released after a read-write personalization, then the IPC connects to the IPS, is authenticated, and writes back the modified portions of the user data and meta data. For a release after a read-only personalization, the IPC simply removes the user's personal data from the device. Note that an IPS can typically only serve user data and meta data for devices that share compatible interfaces for personalization (such as POOM for Windows CE based devices, for instance). The actual scope of devices that can be handled by an IPS therefore largely depends on the availability of suitable interfaces for cross-hardware and

cross-platform personalization as discussed in Section 8.2.2.

Figure 8.2.: GUI of the prototypical Instant Personalization Client (from left to right): login screen dialogue, selection of personalization modules dialogue, auto-release notification dialogue

8.3. Discussion

There are a number of benefits and challenges concerning a practical large-scale deployment of an instant personalization infrastructure, which we discuss in the following.

8.3.1. Sharing and Pooling of Mobile Devices

A person can employ several devices of a kind (e.g., one in the office, one in the car, and one at home), and conveniently switch between those devices by means of instant personalization. And since a device is always stripped of the user's personal and potentially confidential information when it is released after use, a user can temporarily lend devices out to friends and strangers alike without having to worry about the protection of private data.

For this reason, the concept of instant personalization is particularly suited for the *sharing* and *pooling* of mobile user devices in general. While handhelds have been a mainstay in the business world for several years, they are recently also adopted on a larger scale in other areas such as hospital environments [BK02] or education [SNB+01, Fal02]. For instance, the University of South Dakota became one of the first universities to implement a full-scale PDA program, giving faculty members an opportunity to study how the devices can be integrated into college teaching and learning [Pet02]. In such environments, the use of handheld devices can greatly benefit from an instant personalization infrastructure: instead of dealing out mobile devices on a per person basis, each device being permanently owned and exclusively utilized by one user, it becomes feasible to provide a shared pool of devices out of which one can pick any device, instantly personalize it on demand, and use it just as long as needed. Such an approach is not only resource-efficient, lowering the number of devices that have to be bought and maintained, but it also

increases the ease and flexibility of device utilization, as a user no longer relies on his or her own personally owned device but is free to use any available device.

Instant personalization of mobile devices is also advantageous in areas where it is inconvenient or prohibited to take along personally owned electronic devices. To protect the privacy of guests in places such as swimming baths, for example, it may not be desirable that guests bring along their own personal handheld devices that might be equipped with a digital video camera. Instead, the authorities could place generic devices for instant personalization at the guests' disposal, at a *pay-per-use* basis, for example. Such a short-time leasing of mobile devices for instant personalization can also be to the benefit of the guests, as the latter no longer have to worry about personally owned devices being stolen while swimming or being damaged when unintentionally exposed to water or sand.

Further, an interesting question is who should be in charge of operating an instant personalization server, and where the server should be physically located. The availability of an instant personalization server may be unsatisfactory if it is located at the user's home, there being affected by power outages, transient failures of the user's network connection, or unskillful maintenance, for example. In this context, a promising option could be to have telecommunications providers offer the instant personalization service bundled with the traditionally provided communication services. Both services go together well, as connectivity is the prerequisite of anytime, anywhere instant personalization.

8.3.2. Bandwidth Requirements

The availability of a sufficiently high bandwidth may pose a challenge for the practical realization of an instant personalization system. Traditionally, when using a permanently personalized device for remote data access, it is only necessary to regularly synchronize that portion of the data which has been modified either on the device or on the remote server, which may significantly reduce bandwidth requirements.

In contrast, the instant personalization of arbitrary blank devices always requires the complete download and installation of all necessary user data and meta data. Consequently, for instantly personalizing devices that involve a potentially large amount of personal user data (such as laptop computers, MP3 players, or digital cameras with large storage capabilities), high-speed network connectivity is necessary for achieving a swift data transfer. Further relevant factors are the cost and reliability of the data transfer.

When focusing on the instant personalization of resource-limited mobile devices, however, bandwidth usually constitutes a minor problem. On the one hand, the amount of accumulated user data and meta data is comparably low (today the space required for plain-text contact entries on a mobile phone or for calendar entries on a PDA, for instance, is typically in the dimension of a few hundred kilobytes and can be further reduced by means of compression techniques). On the other hand, the data rates of available network technologies such as Wireless LAN (11 Mb/s and beyond), Bluetooth (up to 2 Mb/s), or emerging 3G/4G telecommunication networks (up to 2 Mb/s in 3G networks, and 20 Mb/s and beyond in 4G networks) should be high enough for the realization of a reasonably quick data transfer. The estimated duration of an instant device personalization with regard

to different communication technologies, data rates, and amounts of personal user data is displayed in Table 8.1.[1] We can see that – with the expected emergence of higher bandwidth communication networks – the realization of a truly instant personalization in the range of a couple of seconds or even only milliseconds can be achieved even for comparably large amounts of user data. Once the personalization is completed, the amount of data that has to be transferred back to the server during the release operation is normally uncritical, as it is sufficient to only write back the portion of the data which was actually modified. In the case of a read-only personalization, the release operation can even be efficiently performed off-line by simply erasing any personal information from the device. In addition, progressive update propagation schemes as suggested by Lara et al. [dLKWZ03] could help to further reduce latencies as the user may already start using the device before the personal data has been completely fetched from the server.

In the long run, if the development of computer networks advances at the current rate, one may argue that we ultimately find a close to perfect network at our hands, with global coverage, nearly unlimited bandwidth, high stability and minimum delay. Such a development would obviously greatly facilitate the instant personalization of mobile devices. At the same time, the availability of a nearly perfect network could, provocatively speaking, even remove the need for storing personal data locally on diverse devices. It may even provide a boost for the concept of virtual network computing as described by Richardson et al. [RSFWH98], promoting a unified instant remote access to personalized resources residing in the background infrastructure by using dumb virtual terminals for local input and output only.

However, we think that a complete future shift towards network computing is questionable for several reasons. Firstly, in the past, similarly optimistic prophecies regarding an expected breakthrough of the thin-client approach repeatedly proved to be false and unrealistic. This was for technical and economical reasons, even in the face of a considerable increase of communication bandwidth, or simply just because the underlying concept itself consistently failed to meet the customers' expectations. Secondly, we think that it is doubtful that there will ever be such a (close to) perfect network available. Already today network bandwidth for Internet connections or UMTS communication channels, for instance, is predominantly provided on a best-effort basis, as network providers are generally interested in maximizing the traffic load and keeping excess capacities to a minimum in order to keep down costs and to maintain their competitiveness. Consequently, the quality-of-service characteristics of such communication networks should be far from perfect, particularly in peak times. Thirdly, a solution where computations and data processing are performed locally typically scales better and is more robust against interferences or denial of service attacks than a centralized server approach for which a stable network connection and a remote data transfer is required for each device and each single executed operation.

[1] For calculating the amount of typical personal data used with a PDA or smart phone, we took the following data as a basis: 340 contacts at 1.5 KB each on average, 20 tasks at 2 KB each, and 50 future appointments at 1.2 KB each, yielding a total of approx. 600 KB.

Communication Technology	Bandwidth (nominal)	Bandwidth (net)	PDA (∼600 KB)	e-Book (∼2 MB)	Digital Camera (∼20 MB)	e-Mail (∼200 MB)
GPRS (4 time slots)	57.6 Kb/s	48 Kb/s	1 min 40 s	5 min 33 s	56 min	9 h 16 min
GPRS (8 time slots)	115.2 Kb/s	96 Kb/s	50 s	2 min 47 s	28 min	4 h 38 min
UMTS (global cell)	144 Kb/s	144 Kb/s	33 s	1 min 51 s	19 min	3 h 5 min
UMTS (micro/macro cell)	384 Kb/s	384 Kb/s	13 s	42 s	7 min	1 h 9 min
Bluetooth	2 Mb/s	1 Mb/s	4.8 s	16 s	2 min 40 s	26 min 40 s
UMTS (pico cell)	2 Mb/s	2 Mb/s	2.4 s	8 s	1 min 20 s	13 min 20 sec
WLAN 802.11b	11 Mb/s	5.5 Mb/s	870 ms	2.9 s	29 s	2 min 52 s
WLAN 802.11a, Hiperlan/2	54 Mb/s	32 Mb/s	150 ms	500 ms	5 s	50 s
4G Networks	20-300 Mb/s	100 Mb/s	50 ms	160 ms	1.6 s	16 s

Table 8.1.: Duration of instant personalization with respect to typical communication technologies, net data rates, and different amounts of data (contacts, appointments and task list for a PDA or mobile phone; e-books; digital photos; complete mailbox including e-mail attachments)

8.3.3. Trust and Security

In our system, user *identification* is performed by means of a user name, and user *authentication* by means of a secret user password. Both user name and password are transmitted to the server via a secured communication channel (e.g., using SSL [Net96]). Alternatively, the user can choose to identify him or herself conveniently via fingerprint, removing the need for manually typing a user name or for using an extra identification badge or tag. This was a feasible option since the handheld devices we used for prototyping featured a built-in fingerprint sensor.

Initially, we intended to use fingerprints in place of passwords. However, fingerprint sensors only constitute a secure means for user authentication when embedded in trusted hardware where personal fingerprint information is protected from illegitimate access and tampering [HZ04]. As this is not the case with most fingerprint hardware that comes with today's of-the-shelf handheld devices, an impostor may, with moderate effort, bypass the physical fingerprint sensor and insert another user's fingerprint sample (fingerprints are in general easily available from objects a particular user has previously touched – they then only need to be digitized by the impostor and transformed into the typically publicly known format used by the device-specific fingerprint software).

This raises the question of *trust* in general. When a user possesses several devices of the same kind, or if devices are shared in a closed group (e.g., among friends or colleagues), trust is not an immediate concern. However, am I willing to entrust my private data to a device of unknown origin that may have been tampered with and therefore be potentially malicious and untrustworthy? A publicly available device may be spying on me, secretly stealing personal passwords or disclosing confidential information. It is therefore of prime importance that a user is in a position to clearly assert that any given device has not been tampered with and can be considered trustworthy. A promising attempt to tackle this issue is the Trusted Platform Module (TPM) [Tru03] technology promoted by the Trusted Computing Group.

Alternative methods of secure authentication are one-time authentication schemes, using one-time passwords as first described by Lamport [Lam81], or utilizing trusted hardware tokens carried by the user, including smart cards [LC04] or hardware tokens similar to the ones described by Corner and Noble [CN02]. Another possibility for achieving secure authentication is to use challenge-response mechanisms, such as providing distorted facial images of persons known to the user as a challenge for which he or she has to provide the correct names, or asking the user to recognize a known face out of a selection of otherwise unfamiliar faces, as performed by the Passfaces[2] system, for instance.

Another challenge is the protection of data confidentiality with respect to unauthorized recovery of personal user data: confidential user data that was deleted during the release-phase of a temporarily personalized device should not be recoverable, or only at high cost. Gutmann [Gut96, Gut01] describes the problems and potential solutions in greater detail. Here, the availability of a trusted and tamper resistant hardware module (such as TPM) can also be used to protect a user's personal secrets, by providing a secure storage area which can be completely flushed on demand and which cannot be inspected using memory viewing tools, for example.

[2] Passfaces™ by Real User, `www.realuser.com`

8.4. Prototype Implementation

On the client side, we used HP iPAQ handheld devices of the H5450 series with PocketPC 2002 installed. The client software was programmed in Visual C++ for Embedded Ver. 3.0. On the server side, we used an Intel-based desktop computer running Windows XP, and Microsoft Visual C++ Ver. 6.0 for programming. The HP iPAQ H5450 features a built-in Wireless LAN network interface, which we used for connecting wirelessly to the instant personalization server. The PDA also contains a built-in fingerprint sensor. For programming the fingerprint hardware on the iPAQ, we used the Biometrics API[3] which is freely available as part of the iPAQ Pocket PC Developer Program. For the communication between the instant personalization server and individual clients we used TCP/IP connections. The interaction between clients and the server is performed by means of a proprietary protocol that we developed. The protocol mainly concerns user log-on, user authentication, data synchronization, meta data synchronization (user preferences and settings of the IPC), and remote release operations.

We implemented a functional prototype of the instant personalization system, meeting the basic design goals described in Section 8.2. Currently, the user can choose among four different modules for the instant personalization of the mobile device: personal *tasks*, *contacts*, *calendar entries*, and personal *e-mail settings* (for remote e-mail access using IMAP). Figure 8.3 shows an overview of all implemented GUI functionality of the prototype system. The arrows indicate GUI dialogue transitions that are triggered by user input or external events.

For the implementation of the personalization of tasks, contacts, and calendar, we used the Pocket Outlook Object Model (POOM) as a standardized means of accessing personal user settings and user data on the handheld device. For the adjustment of the e-mail client settings, we had to directly manipulate the Windows CE Registry where all the data about applications, drivers, user preferences, etc. are stored. Authentication is performed by means of user name and password. The integration of the fingerprint sensor (especially the processing of fingerprints on the server side) is still in an experimental stage. Another open task is the integration of compression and synchronization techniques into the module manager to reduce the communication load during data transmission.

Future work should include a user-based evaluation of the system, and focus on the investigation and implementation of the personalization profiles described in Section 8.2.2 in order to support a more diverse set of handheld user devices.

8.5. Conclusion

Mobile user devices such as mobile phones or PDAs are proliferating in everyday life, turning into basic *commodities* that are no longer exclusively sold by specialist stores only, but increasingly offered in supermarkets and fashion stores alike.

As mass-produced handheld devices become available in large quantities and at moderate prices, the concept of instant personalization of mobile devices presents an opportunity to reduce the dependence on single personal devices we permanently possess in response to the threat of hardware faults and inaccessibility faults. In-

[3] `http://devresource.hp.com/drc/technical_papers/Bioapi.jsp`

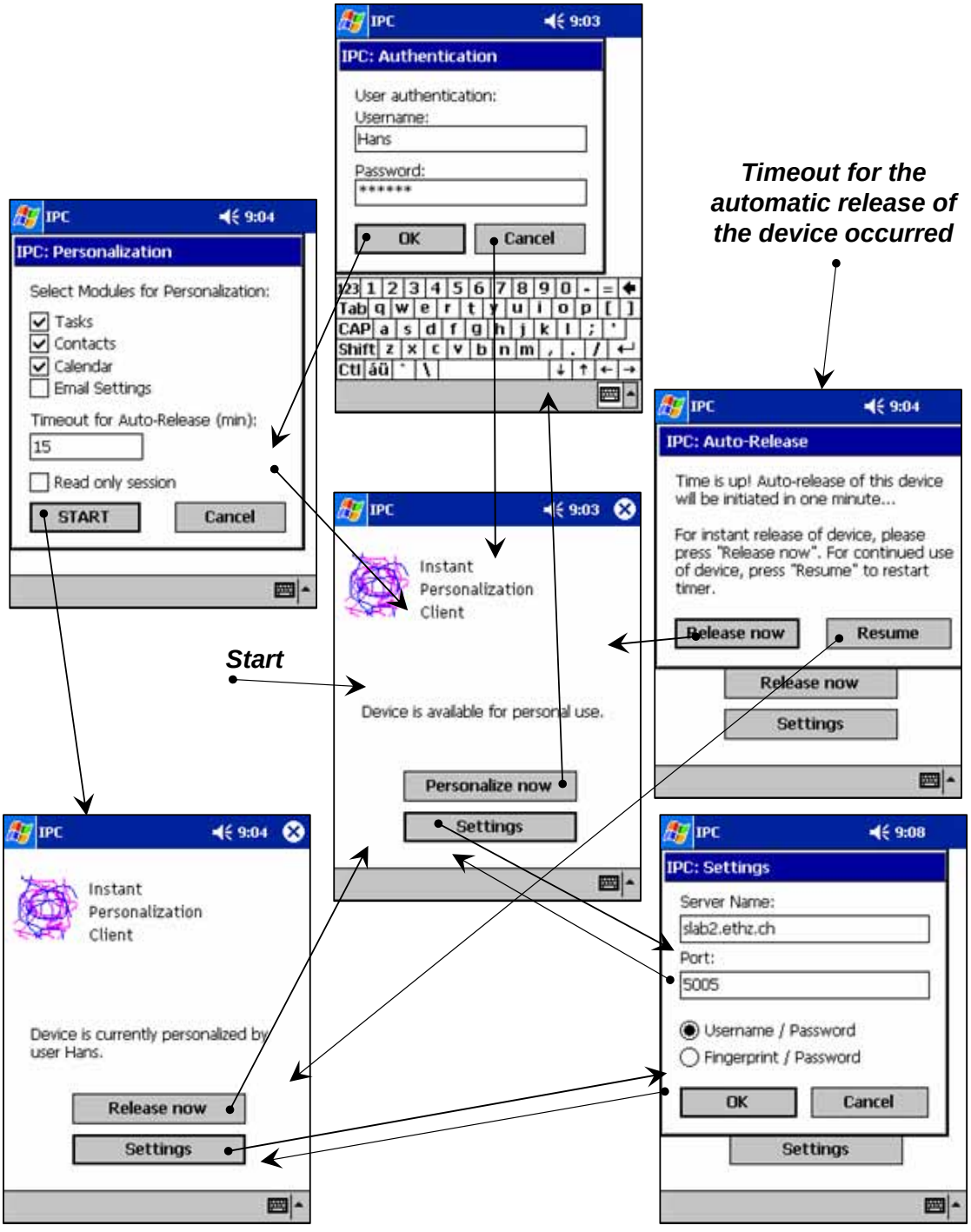

Figure 8.3.: Overview of the implemented functionality and sequence of user dialogue windows of the instant personalization prototype application

stant personalization can help to increase the accessibility of specialized functionality provided by personalized handheld devices, improve the availability of personal user data, facilitate periodic data backup and recovery, and support data confidentiality when devices are lost or stolen.

Finally, with an instant personalization infrastructure in place, end-users are provided with an easy-to-use and efficient tool for prolonging the life cycle of personal user data and meta data, which often outlive the life cycle of the physical devices that carry that data. Instant personalization empowers the user to instantly migrate user data and customizations to substitute devices, even in cases where the original device breaks beyond repair or gets lost or stolen.

8.6. Related Work

In this chapter we presented the goals and requirements of instant personalization and temporary ownership for mobile user devices, and described an initial prototype we developed that supports our core concepts on Windows CE devices.

Our work is closely related to the research domain of *ubiquitous data access*, where the goal is to achieve reliable *anytime*, *anywhere* access to user data by means of heterogeneous devices.

A prominent approach for ubiquitous data access is the *UbiData* system by Zhang et al. [ZHH03], an application-transparent middleware architecture which provides device-independent access to data from heterogeneous sources. Here, device independence relates to the fact that the system allows the user to switch among his or her various personal devices (such as the personal office PC, laptop, and PDA). Typically these devices are permanently owned and personalized by the user.

In contrast, we are suggesting a diversification of access to personalized device functionality and user data by enabling the user to pick any *impersonal* handheld device of a certain type, thus considerably increasing the choice of devices from a small number of personally owned to a potentially unlimited number of available devices. Further, different from our work, Zhang et al. focus on issues of automatic and device-independent selection, hoarding, and synchronization of data, but they are not supporting an instant and temporary personalization of arbitrary mobile devices. In our approach, instant personalization gives the user not only access to the specific user data he or she normally uses with a certain type of device (e.g., personal calendar for the PDA, personal address book of the mobile phone, etc.), but also temporarily installs the specific meta-data that is required for the proper and convenient functioning of the characteristic services and applications provided by the particular type of device (including customizations, application settings, passwords, etc.). Another advantage of our approach is that it implicitly provides the end-user with support for device/data life cycle management by transparently and instantly performing the migration of the user's personal data and meta data. In our approach, we provide additional *control options* to the user in order to preserve the confidentiality of personal user data with regard to devices that are physically unavailable (e.g., due to inaccessibility faults as described in Sect. 4.1.3). These control options include a manual, timer-controlled, or user initiated and server-mediated release of physically absent devices, upon which all confidential user-related data is wiped from the device. This approach is inherently different from the meta data and profile management in [HH04], where meta data is primarily

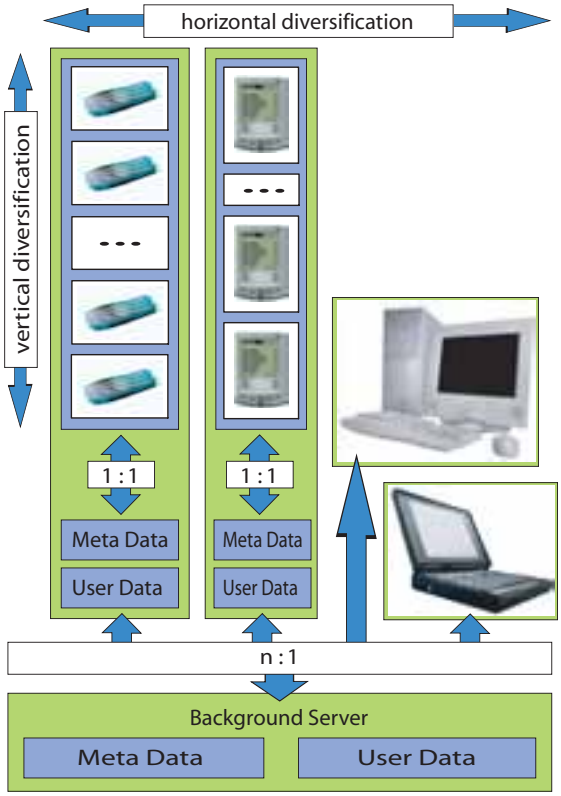

Figure 8.4.: Horizontal and vertical diversification of data and device access

used for building so-called converged networks in which users can freely use a host of services from cooperating providers through a single sign-on.

However, the diversification of device access by means of instant personalization would blend well with ubiquitous data access mechanisms such as used in the UbiData system, as the first could help to significantly reduce the dependence on individual personally owned and therefore permanently personalized devices of a kind. While a person may concurrently use n different types of devices as part of a "horizontally" diversified ubiquitous data access, each of these device types can in return be "vertically" diversified by enabling the user to instantly load his or her device-specific portion of user data and meta data onto any device out of a virtually unlimited number of devices of the same kind (see Fig. 8.4).

Want et al. proposed a different approach to achieve ubiquitous data access in the face of user mobility. They use a portable mobile storage device enhanced with wireless communication capabilities, the *Personal Server* [WPD+02], which enables nearby devices to get access to the user's personal files. Currently, the Personal Server does not support the instant personalization of mobile devices, but it could in principle be used as a local personalization server. However, compared to our approach, the Personal Server constitutes a single point of failure, and in this respect it suffers from the same shortcomings as any individual mobile device a user owns and carries along: if it breaks down beyond repair, or in case it is lost or stolen, all the data on the Personal Server that has been modified since the last backup is definitely lost. And if the Personal Server is only temporarily unavailable (e.g., battery is depleted or the user unintentionally left the device behind), the user has no means of accessing his or her personal data on the spot, either.

There has already been a significant amount of research in the field of personalization of services in ubiquitous computing environments [CJ02, LvKSP02, SMPC02, HT04]. However, here the main focus so far has been on providing personalized services to mobile devices (such as personalized content delivery, content and service adaption, and personalized interfaces for interaction with nearby devices, for instance) instead of instantly personalizing the mobile devices themselves.

The Microsoft Active Directory service for Windows-based personal computers supports user mobility within a distributed computing environment inside of an organization. It provides a single-log-on capability and a central repository for information, simplifying user and computer management and providing access to networked resources within a Windows domain. Compared to our work, the Microsoft Active Directory is a heavyweight infrastructure focusing on centralized user and computer management, while our approach is lightweight, targeting resource-limited handheld devices, increasing the accessibility of device functionality and user data, and protecting user data on personalized devices that are lost or left behind by means of a server-triggered or timeout-triggered data recovery mechanism.

Further related is the field of network computing. Here the main idea is to utilize a thin client with basic input/output capabilities to control applications executed on a remote computer. One example hereof is the Remote Desktop technology by Microsoft for Windows-based computers. Another example is the Virtual Network Computing (VNC) system [RSFWH98], which provides access to home computing environments from anywhere and any device via a network connection by using a simple platform-independent display protocol. In contrast to our work, personalization of mobile devices is generally not an issue in network computing, as the latter is rather aiming at providing an abstraction from the underlying device hardware to achieve device-independence. As a result, the network computing approach is not suited to exploit particular device-specific functionalities of mobile user devices. Moreover, a thin client usually requires a permanent network connection while controlling a service on a remote computer, whereas in our approach, a mobile device can operate autonomously and without requiring a network connection once it has been personalized, thus being unaffected by transient disconnections or network delay, for instance.

Acknowledgments

The author wishes to acknowledge Lukas Stucki for his work on the implementation of the instant personalization prototype [Stu04].

Part III.

Dependability in Context-Aware Computing

9. Concepts for Dependable Context-Aware Computing

In this chapter, we elaborate two concepts for enabling fault-tolerant context-aware computing based on localized cooperation and resource sharing, and on redundant sensor fusion. The concept of *super-distribution of smart entities* enables the provisioning of dedicated redundancy that can be exploited for fault-tolerant location-aware services based on localized resource sharing. Further, we present two systems for the dependable self-positioning of mobile devices based on *redundant sensor data fusion*.

9.1. Dependable Context-Aware Computing Through Fault Tolerance

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 [DA00]. Furthermore, context can be classified into different categories: *computing context* (locally available resources, network connectivity, communication bandwidth, etc.), *physical context* (physical properties of the user's environment, such as temperature, humidity, lighting, noise level, etc.), and *user context* (e.g., the user's profile, location, nearby people, the user's activity, and so on) [DA00].

Accordingly, fault-tolerance mechanisms for achieving dependable context-aware computing can be categorized according to the type and complexity of context they are associated with.

9.1.1. Fault-Tolerant Operation by Means of Localized Cooperation and Resource Sharing

In a ubiquitous computing environment, the local resources that are available as part of a user's computing context constitute a valuable source of *localized ad hoc redundancy*. Therefore, computing context presents itself to be exploited for the realization of fault tolerant applications and services executed on mobile devices based on *localized cooperation and resource sharing*. For example, a mobile device could use its computing context to overcome transient memory space bottlenecks by temporarily storing data on other physically proximate devices.

9.1.2. Fault-Tolerant Context Sensing

Physical context is determined by the physical properties of the user's environment. So physical context refers to the immaterial, qualitative state of the user's

immediate environment, whereas the computing context is concerned with a material, quantitative assessment. Here, fault-tolerance mechanisms typically can be applied to the process of *sensing of physical context*. Assuming that a particular physical context can be sensed accurately and effectively by applying a single sensor technology (e.g., using a mechanical sensor that directly measures a physical phenomena), fault tolerance can be achieved by employing classic redundancy, such as by duplicating or diversifying sensors and using a basic voter unit that determines the presumably correct result. For instance, the sensing capabilities of a smart portable fire detector can be rendered fault-tolerant by using a set of diverse, redundant sensors that measure smoke concentration, temperature, and infrared light intensity – then the failure of any two individual sensors can be tolerated as long as the remaining sensor functions correctly.

User context generally is concerned with information about the actual activity, intention, or situation of a user. Like in the case of physical context, some user context can be directly determined through suitable sensors without the need of complex processing, such as position (with regard to a coordinate system), acceleration, and orientation of the user, for instance. We call this user context *basic* user context. Here, established fault-tolerance methods such as replication and diversification of sensors can also be applied to increase the reliability of the sensing process. Hence we will not consider fault tolerant context sensing any further as part of this dissertation.

9.1.3. Fault-Tolerant Data Fusion and Context Inference

The computation of more complex, higher-level user context usually involves the drawing of conclusions from different available sources of context information. Typically this includes both the current computing context, physical context, and basic user context, as well as the application of problem-specific models and/or knowledge bases to enable semantic interpretation and reasoning.

For instance, to infer that a user intends to print a document on his PDA on a physically nearby printer at which the PDA is pointed at upon selecting the print command in the active word processing software, several different pieces of context have to be combined. For instance, required context knowledge could include (1) the ID and (2) position of the printer, (3) the deliberate orientation of the PDA over the last few seconds, (4) knowledge on how to infer and interpret the pointing gesture (e.g., the orientation of the PDA has to be maintained for a significant amount of time within certain boundaries), (5) the currently active document, (6) the communication interface of the printer, and (7) the issuing of the print command through the user.

To render the process of *information fusion and context inference* fault-tolerant, it has to feature redundancy of some kind, such as sensor hardware redundancy, quality-of-service redundancy, or functional redundancy (cf. Sect. 4.5). For instance, given the earlier Mediacup example of Sect. 3.2.3, a simple semantic model for detecting a meeting (activity context) is that there are *at least two moving cups filled with warm coffee in the same room with closed doors*. The inference of this higher-level context can be made fault-tolerant by means of a *redundant information fusion and inference algorithm*, with the goal to tolerate the inaccuracy or unavailability of individual pieces of information. Concretely, the fusion algorithm

could consider further redundant sources of context information, such as sensors in the room that detect the noise level, movement, temperature variations, or pressure sensors embedded in chairs or in the floor, and so on.

9.2. Concepts for Fault-Tolerant Operation By Means of Localized Cooperation and Resource Sharing

By enabling the discovery and usage of external resources of physically proximate devices as part of the local computing context, smart objects and devices can perform localized cooperation and collaboration in order to make up for their limited resources [OMG04, SFV04].

Redundancies with regard to resources found in the computing context of a device ideally can be tapped and exploited as localized ad hoc redundancy for enabling fault-tolerant computing (see Sect. 4.5.5). By allowing the temporary use of unallocated resources of nearby devices, mobile devices can be put into a position to temporarily tolerate transient resource bottlenecks or component failures. Resource-limited mobile and handheld devices in particular benefit from such a localized resource sharing, as for them such a procedure may significantly increase the number and diversity of accessible resources.

We distinguish two types of redundant user-centric resources that can be found in the local computing context of a mobile user device, and which differ with regard to constancy and control: *volatile* redundant resources and *dedicated* redundant resources. The utilization of these redundant resources for fault-tolerant context-aware computing requires different approaches and considerations, as we explain in the following.

9.2.1. Volatile User-Centric Redundancy: Cooperative Smart Everyday Objects

Volatile User-Centric Redundancy

The numerous diverse smart objects and devices that are part of the computing context of a mobile device in a ubiquitous computing environment provide *volatile user-centric redundancy*: they form a pool of volatile user-centric devices and objects whose resources are redundant in numbers and diverse in terms of technology and functionality. The most basic technical resources these devices share are processing capabilities, memory storage, and one or more communication interfaces such as wired local area networks (prevailingly for stationary objects), Wireless LAN, Bluetooth, and ZigBee. As devices may be carried on and/or be controlled by individual users, many resources within the vicinity of a context-aware device show random characteristics: they are liable to come and go spontaneously, and form an ad hoc composition [GDL+04] of services, applications, and devices at runtime with a high volatility of cooperative relationships and topologies (cf. Section 3.3.10). Cooperation therefore prevailingly takes place between random partners in an opportunistic and unpredictable fashion.

Fault Tolerance Based on Volatile User-Centric Redundancy

Mobile devices can make use of volatile user-centric redundancy in order to tolerate the (transient) unavailability or undersupply of resources (such as memory, energy, connectivity, for instance). In particular, individual user devices can harness their local volatile computing context for the realization of localized fault-tolerant operation. The locality aspect of additional resources used for fault-tolerance purposes is important since users often depend on the operability of a particular personal mobile device (which we call *primary* mobile device) for the carrying out of a special task. If the proper operation of a primary mobile device is threatened by faults, fault-tolerance mechanisms have to acquire any required redundancy from the computing context at the current location. This is in contrast to system-centric systems such as sensor networks where single devices may be sacrificed for the realization or optimization of overall system-wide goals.

Fault-Tolerant Mobile Applications By Using Cooperative Smart Everyday Objects

Due to the unpredictability of ad hoc acquired external resources with regard to availability and delivered quality of service, the realization of fault-tolerance mechanisms for mobile applications based on volatile user-centric redundancy is only meaningful if the additional uncertainty about resources does not outweigh a potential gain.

To address this challenge, we developed a system that enables applications on mobile handheld devices to tolerate a number of faults by exploiting their local volatile computing context. The idea is to empower individual mobile devices to cooperate with physically nearby devices by means of a device-abstraction (called "smart object") and a distributed middleware layer that provides support for fault-tolerant data transfer and communication services. The concept of a smart object results in a device-independent service interface that enables ad hoc cooperation and resource sharing among physically proximate and possibly heterogeneous devices.

In Chapter 10, we describe our fault-tolerance middleware layer based on a smart everyday object infrastructure in more detail. We further present a prototypical reference implementation of a fault-tolerant and adaptive patient monitoring system that makes use of our middleware.

9.2.2. Dedicated User-Centric Redundancy: Super-Distribution of Smart Entities

Dedicated User-Centric Redundancy

Rather than to rely on volatile resources that in the worst case may not be available in sufficient quantities, computerized entities can be deliberately distributed in a dense, stationary, and redundant fashion over floor and object surfaces to provide an area-wide infrastructure of dedicated, densely distributed resources. While the dense distribution of such *static dedicated entities* causes additional costs for deployment and maintenance, it has the advantage that these resources can be taken for granted and controlled by other devices. Furthermore, the immobility

of the distributed entities makes them suitable for persistently storing information that is characteristic for the respective physical location. This information could then be directly retrieved by other devices in an ad hoc fashion without need of infrastructure-based connectivity.

Fault Tolerance Based on Dedicated User-Centric Redundancy

Naturally, static dedicated resources distributed in the environment provide a more predictable means of redundancy than volatile resources do. Specifically, static dedicated resources found in the immediate locality of a mobile device as part of its computing context can be used in an ad hoc fashion to extend the limited resources of that mobile device. In addition, such static dedicated resources that are physically distributed can further be used as local carriers of location-dependent context information. This helps to increase the flexibility, autonomy, and fault tolerance of mobile devices, as the dependence of the latter on centralized background infrastructures is reduced in the process.

In order to enable independent, autonomous devices to benefit from dedicated redundancy, an important challenge is to provide open middleware architectures and interfaces to these resources, thus respecting the open-world assumption of ubiquitous computing environments to support interoperability and interconnectivity (cf. Section 3.3.9).

Fault-Tolerant Location-Aware Computing by Means of Super-Distributed Smart Entities

We have investigated means of using dedicated user-centric redundancy for fault-tolerant location-aware computing. In the process, we came up with an approach that employs densely distributed computerized (and therefore "smart") entities in the environment for the realization of a fault-tolerant service middleware. The resulting physical infrastructure we call a super-distributed smart-entity infrastructure. The term *super-distributed* refers to the characteristic that the smart entities are distributed in a highly redundant fashion.

The investigation of the concept of super-distribution constitutes a central contribution of this dissertation. Due to our extensive amount of work in this area, we provide a more detailed overview of our investigated research issues and results in Sect. 9.4.

9.3. Concepts for Fault-Tolerant Data Fusion and Context Inference

In ubiquitous computing environments, devices may passively interact with their surroundings by sensing nearby resources and signals. For instance, mobile devices may sense physical properties such as temperature and humidity, or the presence and signal strength of radio transmitters, to infer knowledge about their current place, environmental condition, and contextual situation, or about the activity and intention of their respective users.

The dependability of the context sensing process is determined by the robustness of the applied data fusion and context reasoning mechanisms, and by the dependability of the physical sensing components (*sensors*) that provide contextual input.

Independent from the expressiveness and quality of a particular applied fusion procedure, the robustness of its reasoning mechanism generally depends on its ability to integrate redundant sensory information. Due to the complexity of the reasoning procedures and the ambiguity of sensory information obtained from individual sensors, the inference of context often requires the combination of diverse sensors to improve the certainty of the reasoning [SBG99, GSB02]. If the diverse sensors available for reasoning are non-redundant, the robustness of the reasoning itself is low as the failure of individual sensors may lead to utterly false conclusions. For instance, to determine if a person is drinking from a cup, both the information that the cup is filled with a liquid and that it is lifted and tilted has to be combined. However, if each piece of information is analyzed separately, there is a significantly higher probability of a false reasoning. For instance, a cup may be left behind at a table even though it is still filled to some level, or a person may just be playing with an empty cup, turning and tilting it in the process.

9.3.1. Redundant Multi-Sensor Data Fusion

By fusing *redundant* sources of sensory context information, it is possible to improve the *reliability* and the *availability* of a context-aware service. A redundant sensor data fusion mechanism can use (1) multiple sensors of the same kind to compensate for individual hardware failures (homogeneous hardware redundancy, see Sect. 4.5.1), or (2) diverse sensors (heterogeneous hardware redundancy or functional redundancy, see Sections 4.5.2 and 4.5.7, respectively) to cope with adverse environmental conditions that only render single types of sensors inoperative (e.g., transient interference caused by a strong disturbing signal disrupting only radio-based sensors while auditory or visual sensors remain unaffected), (3) or combine both homogeneous and heterogeneous redundancy at the same time. With regard to ubiquitous computing environments, an important challenge is to identify and harness the multitude of diverse devices and technologies found therein for context sensing and derivation. In particular the utilization of natural, commonplace sources of physical context information, such as light, temperature, humidity, and sound, should be considered in the process, as the advances in micro-computing and ubiquitous computing technology finally make it possible to integrate various miniature sensors even with everyday objects and devices.

The *quality* of a context-aware system to a large extent depends on the quality of the applied information fusion and reasoning mechanisms, which are often specific for a particular problem or application domain. If a context-aware system is redundant with regard to its achieved quality of service (e.g., it over-fulfills the effective quality-of-service requirements), it may be able to reduce its delivered quality of service in order to tolerate the temporary unavailability of sensory input on behalf of some of its sensors (qualitative redundancy, see Sect. 4.5.6). The study of the expressiveness and validity of context fusion models on the semantical level and of the corresponding inference mechanisms that enable to draw human-understandable conclusions from sensory context data is a wide field of research and beyond the scope of this dissertation.

But what such context inference systems have in common is that, for achieving dependability, the development of the applied context fusion algorithms should be fitted with the capability of combining *redundant* sources of context. This is in particular important to increase the reliability and availability of mobile applications that need to directly determine context in situ without the support of a background service infrastructure. Potential context fusion architectures for ubiquitous computing should therefore be simple and efficient enough to be also executed on portable, resource-limited devices such as mobile phones and PDAs, which are currently increasing massively in numbers (cf. property *ubiquity* in Section 3.3.3 on page 18).

9.3.2. Case-Study: Fault-Tolerant Positioning with Quality-of-Service Guarantees by Means of Multi-Sensor Data Fusion

We developed a robust positioning system (called iPOS) that employs *multi-sensor data fusion* for achieving fault tolerance and adaptability. In the process, we applied an open data fusion architecture that is lightweight enough to support small, resource-limited mobile devices (here: personal digital assistants). The data fusion architecture is extensible in the sense that new position sensing technologies as well as arbitrary third-party positioning services can easily be integrated with the system. Further, by using a map model, the system is capable of integrating both symbolic or geographic representations of position information, as well as local and global geographic position coordinates. A special feature of the fusion algorithm is that it is able to provide quality-of-service guarantees (QoS guarantees) under certain conditions. A description of the iPOS positioning system and its data fusion architecture is presented in Chapter 14. There we also present a prototypical reference implementation of the iPOS system, which we evaluated by means of practical experiments.

9.3.3. Case Study: A Self-Calibrating Real-Time Positioning System Based on Redundant Sensor Fusion and Context Awareness

The two concepts of *localized cooperation and resource sharing* and *redundant sensor data fusion* can also be combined to improve the dependability of a single system.

We developed a hybrid system that is capable of combining a solar-cell based positioning technique with a positioning service that makes use of local cooperation and interaction with a super-distributed smart-entity infrastructure. Firstly, the fusion of the two redundant positioning techniques enables mobile devices to tolerate areas where one system fails. The redundant positioning capabilities can further be exploited for increasing energy efficiency by selectively using the system with the lowest power requirements. Secondly, by making use of location-dependent context information provided by a super-distributed RFID tag infrastructure, the accuracy of the calculated positioning estimates of our implemented prototype system could be significantly improved by approx. 20% from 28 cm to 22 cm in 80%

of all calculated estimates. A detailed description and experimental evaluation of the hybrid system is presented in Chapter 15.

9.4. Super-Distribution of Smart Entities as a Design Principle for Fault-Tolerant Location-Aware Computing

In the following sections, we introduce the concept of *super-distribution of smart entities* as a design principle for fault-tolerant location-aware computing.

9.4.1. Super-Distribution as a Design Principle

Location information has become increasingly important for various mobile and portable devices, opening diverse application fields such as travel information, sight seeing, shopping, entertainment, event information, education, and health care [MB03]. It can be used to provide the user with location-aware services whose execution can be dynamically adapted to the characteristics of the user's particular context at his or her respective location [CLMZ03]. At the same time, as a result of ongoing advances in miniaturization and micro-computing, it has become feasible to distribute small computerized entities in large quantities over object surfaces [BM04].

Based on these observations, we motivate *super-distribution* as a design principle for the realization of reliable and highly available location-dependent services for mobile devices. The main idea is to distribute small computerized physical objects in large quantities over object surfaces, such as across floor spaces or walls, to obtain a dense and highly redundant distribution of "smart entities". The resulting smart entity infrastructure then forms the basis for the development of a *fault-tolerant middleware* that provides a set of fundamental location-aware services and abstractions. In return, this middleware serves as a foundation for the development of dependable location-aware mobile applications.

We chose the term super-distribution in allusion to *super distributed objects*, a notion that refers to the large quantities of computerized devices and software objects equipped with communication capabilities that are typically found in ubiquitous computing environments [OMG04]. One of the research goals in the field of super distributed objects is to provide a generic middleware that enables such objects, which may differ significantly in terms of hardware and software capabilities, to cooperate and interact, forming a vast heterogeneous smart object infrastructure [FKSK02, SVG+03]. In this context, the term "super distributed" aims at describing a de facto state or quality of the environment, which is characterized by a dense and redundant distribution of various heterogeneous (and mobile) devices. In our case, however, we place particular emphasis on the distribution itself, and on the qualities of the resulting (stationary) infrastructure that we are able to shape in the process.

9.4.2. Super-Distribution of Smart Entities

We define a *smart entity* (SE) as a physical artifact that is enhanced by embedded computing technology of some kind. The minimum requirements we demand of a SE are the following: it has a globally unique identifier and a built-in memory with data read/write capabilities, both of which can be accessed by physically nearby devices via wireless ad hoc communication. Examples for smart entities are simple radio frequency identification (RFID) tags as well as self-powered embedded sensor nodes.

We define *super-distribution of smart entities* as the process of deploying and distributing smart entities in a dense, highly redundant (or *abundant*) fashion. We call the resulting infrastructure a super-distributed smart-entity infrastructure (SDI). In this context, dense means that a mobile device that moves within an SDI will always find other physical smart entities within ad hoc communication range for in situ interaction. Highly redundant means that the degree of redundancy is not deliberately set to a fixed value, but determined by an abundance of entities found in any physical location of the infrastructure. *Abundance* refers to a quantity of entities that significantly outnumbers the amount of entities that, ideally, would be required for a non-fault-tolerant, non-redundant operation of SDI-based services in the absence of disturbances.

In the following, we only consider a *stationary* super-distribution where the individual smart entities are permanently attached to a substrate (e.g., floor spaces and object surfaces) at a physical place that is well-defined within a local or global coordinate system. Of course, smart entities can also be super-distributed on the surface of mobile physical objects, such as on a table, within a car, or on the floor space of a boat – in these cases, these smart entities are stationary with regard to their respective local environment.

From the user's perspective, interaction with the SDI is performed by means of a *mobile device* (MoD). The MoD features a wireless communication interface, which enables it to communicate in an ad hoc fashion with smart entities in its immediate vicinity. On each MoD, an independent instance of the SDI service middleware is installed and executed. A MoD can be carried by a user, or it may be part of other devices, such as being integrated into a vehicle or into a blind man's stick, for example.

9.4.3. Range of Application

In the following, we describe a number of potential application fields that highlight some opportunities and advantages of an SDI-based services from the perspective of the user and systems designer.

User-Centric Location-Dependent Services

Within an SDI, users directly interact with the smart entities at their current location. For geographic guidance and navigation, a MoD can determine its current position by calculating an estimate from the individual positions stored on nearby smart entities, or simply look up the current location with the help of a local map containing the positions of individual smart entities. Users are able to share information about local points of interest or leave personal messages directly in

the physical places where the information is most helpful and required. Public directories, whose entries are physically distributed across the smart entities of the SDI, provide localized information about room numbers or names of departments, offices, or personnel, enabling users to find their way unassistedly even in unfamiliar places and buildings. Besides, users can leave virtual data traces in an ad hoc fashion on the smart entities passed along the way, permitting friends or colleagues to follow at a later point in time to places where an activity or meeting is to take place. By integrating the MoD into a blind man's stick or into the underside of a wheelchair, the described services can also be made available to visually impaired people or persons with walking disabilities. This further enables these users to share information tailored to their particular needs in situ, such as information about nearby obstacles, dangerous crossings, handicapped accessible ramps and gangways, etc.

Dependable and Safety-Critical Services

Adverse conditions or physical damage caused by natural or human factors (e.g., earthquakes, heavy weather, fires caused by arson, or terrorist attacks) often lead to the failure of infrastructure services in buildings or public places, disrupting electricity, landline telephony, communication networks, and networked computer infrastructures. However, by maintaining safety-critical information in smart entities at the physical places where it is required, SDI-based services remain operational in physically intact areas of the SDI even when conventional background service infrastructures collapse or other areas are damaged, as the MoD directly interacts with local smart entities via short-range ad hoc communication. In addition, an SDI allows for the provisioning of dedicated emergency services. For instance, by means of permanent virtual data traces stored on the smart entities of an SDI, it is possible to provide services that direct users (including professionals such as firefighters and emergency physicians, or persons with impairments) to the nearest emergency exit or life saving equipment.

Systems Support for Collaborative Activities

In some cases the activities of individual MoDs can be combined to contribute to a superordinate task. For instance, MoDs that have a third-party positioning service at their disposal can store obtained position readings on the smart entities at their respective places. Thus the SDI can be "bootstrapped" with position information over time through a collaborative effort. Another example for collaboration is the construction of a global site map of an SDI by joining partial SE mappings obtained from individual MoDs. Further, an SDI can serve as a vast communal information space: the individual contributions of users in different physical places can contribute to the creation of open, community-driven information services and directories, such as communal shopping-, restaurant-, and city-tour-guides, for instance.

9.4.4. Radio Frequency Identification as Enabling Technology

As an enabling technology for smart entities, we decided to used radio frequency identification (RFID) tags for technical, practical, and economical reasons.

Firstly, RFID tags fulfill the minimum requirements that we demand of smart entities: they possess a unique identifier and memory storage capabilities.

Secondly, owing to a rapid proliferation of RFID technology, RFID hardware including RFID reader devices, antennas, and tags have become increasingly smaller and cheaper. As a result, the deployment of RFID technology on a larger scale is about to become both technically and economically feasible. Hitachi, for instance, is about to commence mass production of the *mu-chip* [Hit06], which is a miniature RFID tag with a surface area of 0.3 mm^2. Further, the Auto-ID Center has proposed methods which could lower the cost per RFID chip to approx. five US cents [Sar01].

Thirdly, the large-scale deployment of RFID tags has not been deeply researched and poses interesting challenges. In the conventional process of RFID tag deployment prevailing today, only a limited number of passive tags are placed in the environment in a deliberate and sparse fashion. Typically, RFID tags are mainly used for identifying objects [YHAP02, CR03] and for detecting the containedness relationships of these objects [LF04]. Explicitly placed stationary tags embedded in the environment also serve as dedicated artificial *landmarks*. They can be detected by means of a mobile RFID reader and are used to support the navigation of mobile devices and robots [KBPD97, NLLP03, PFF+03], or to mark places and passageways [GHM99].

In contrast to conventional means of RFID tag deployment and utilization, we developed the concept of *super-distributed RFID tag infrastructures*, which we present in detail in Chapter 11. Based on the concept of super-distribution, we advocate *massively-redundant* tag distributions where cheap passive RFID tags (i.e. tags without a built-in power supply) are deployed in large quantities and in a highly redundant fashion over large areas or object surfaces. In doing so, we showed that the identity of a single tag becomes insignificant in exchange for an increased efficiency, coverage, and robustness of the infrastructure thus created as a whole. We further demonstrated that such an approach opens up a whole spectrum of possibilities for creating novel RFID-based services and location-dependent applications, including a new means of cooperation between mobile entities. We also discussed some of the technological opportunities and challenges, with the intention of stimulating further research in this area.

9.4.5. Fault-Tolerant Service Middleware

Using our results with regard to the super-distribution of RFID tags as a starting point, we developed a middleware architecture based on super-distributed smart entities in general. Based on an analysis of fundamental location-aware services that may be supported by a super-distributed smart-entity infrastructure, we worked out a layered service middleware architecture. The middleware was designed to harness the dedicated localized redundancy of the super-distributed smart-entity infrastructure for achieving fault-tolerant operation of the provided services and applications.

In the service middleware, dependability and fault-tolerance aspects play an important role at two different levels of abstraction: Firstly, the middleware services themselves had to be robust and fault-tolerant. On the one hand, they had to be able to tolerate failures of individual smart entities during operation. On the other hand, the middleware services had to be capable of integrating newly distributed smart entities with minimal user intervention, to achieve a high serviceability (i.e., the ability to undergo maintenance without shutting down the system), and to minimize maintenance overhead. Secondly, the middleware should enable the development and the operation of reliable and highly available location-aware applications. Here, the main goal was to exploit the *locality* aspect of a super-distributed smart-entity infrastructure in order to provide basic services in situ that still function in the absence of network connectivity or failure of a background computing infrastructure.

The design of the middleware and the description of its fault-tolerance mechanisms are presented in detail in Chapter 12.

9.4.6. Prototypical Reference Implementation

To assess the practicability and effectiveness of our middleware architecture, we prototypically realized several exemplary services and applications. For the construction of the prototypical smart entity infrastructure, we chose radio frequency identification (RFID) as a technology, the suitability of which we already discussed earlier. Thus the smart entities were represented by passive RFID tags, and the Hardware Layer in our prototype implementation constituted a super-distributed RFID tag infrastructure as described earlier in Chapter 11. The reference implementation based on RFID technology is presented in Chapter 12.

9.4.7. Motivating Scenario

By densely distributing computerized entities over object surfaces, it is possible to obtain a high degree of redundance and abundance of localized resources in physical places. The following scenario explains how such super-distributed smart-entity infrastructures enable the development of fault-tolerant context-aware services and applications for the average user, minority groups, and specialized professionals alike.

> John is a visually impaired person. He relies on a blind man's stick for walking, which features an integrated radio frequency identification (RFID) reader and an RFID antenna at its tip. The RFID reader is wirelessly connected to a personal digital assistant (PDA) that John carries in his pocket. The PDA processes location-dependent data read from RFID tags that were detected with the blind man's stick in order to provide various location-aware services. For instance, any obtained navigation and local directory information is delivered to the blind user via an audio interface.
>
> Today John wants to collect his new passport at the registry office situated in the town hall, at the desk of a Mr Doe. The corridors of the public town hall building are outfitted with carpets containing a dense

distribution of RFID tags. The building further features a Wireless-LAN-based positioning system. As John enters the building, his PDA informs him that he now has a WLAN-based navigation service and an RFID-based personnel directory and emergency service at his disposal. Guided by the WLAN-based navigation system, he reaches the vicinity of the registry office. Now he selects the RFID-based personnel directory, which enables him to walk along the corridor while obtaining constant feedback about the titles and the names of the personnel of the offices he passes. He finally reaches the door to Mr Doe's office, when a gas explosion shakes the building and knocks John off his feet. After some time, John recovers from the blow and gets up, smelling smoke from fire. Unfortunately the WLAN-dependent navigation system is no longer responding as the explosion knocked out the electricity and computing infrastructure of the building. However, John is relieved to find that the RFID-based emergency service is still active: a virtual data trace stored on the tags of the RFID-tagged carpet leads the way to the nearest emergency exit. Using his blind man's stick as a detector, the PDA continuously informs John that he is getting closer to the exit as he moves along the corridors. As John takes the wrong turn at a fork, he is warned that he is moving into the wrong direction, away from the exit, upon which John returns to the fork and chooses the correct corridor. At some point John walks upon a carpet that is partially burned so that some of the integrated tags are no longer functional, but fortunately the number of remaining operational tags is still high enough to get frequent tag readings with his blind man's stick. Finally he arrives at the exit where he is already received by staff helping him down the stairs.

In the meantime, the smoke within the building reduced visibility to below one meter. Thomas, a professional firefighter, is told by radio that Ms Clarke of the accounting department is still missing on the second floor. Thomas has a mobile RFID reader and antenna unit integrated with the soles of each of his boots. The RFID reader is connected to Thomas' PDA, on which a visual navigation software is executed. The navigation system uses a map where the IDs of all RFID tags distributed across the corridors of the town hall building are recorded. As Thomas walks along the corridors, the navigation system updates his current position whenever an RFID tag is detected underneath his boots, even in areas where the carpet is partly destroyed. He arrives at Ms Clarke's office door where he encounters fire. Since the fire extinguishers are not part of the map of the navigation system, he activates the RFID-based emergency service called "find nearest fire extinguisher". Then Thomas follows the bright arrows that are shown on his PDA indicating the direction of the nearest fire extinguisher, whose distance is displayed as being only 10 meters away, and carries it back to Ms Clarke's office door. He extinguishes the flames and finds Ms Clarke unconscious nearby her desk, still unharmed from the flames. Following the shortest escape route displayed on his PDA, he rescues her to safety.

10. Fault-Tolerant User-Centric Data Dissemination and Communication Services Based on Cooperating Smart Everyday Objects

In this chapter, we present a fault-tolerant and user-centric service infrastructure based on cooperating smart everyday objects. The main goal is to exploit local ad hoc redundancy for enabling fault-tolerant and adaptive operation of users' resource-limited mobile devices. Concretely, the infrastructure aims at (1) increasing the dependability of data dissemination from mobile user devices to remote addressees, and at (2) improving the connectivity and energy efficiency of mobile user devices by using physically proximate smart objects as communication gateways.

In the following, we first further motivate the use of smart objects. Then we describe the world model and dependability issues of our infrastructure. Finally we present our prototypical implementation and draw some conclusions.

10.1. Dependable User-Centric Computing Based on Cooperating Smart Everyday Objects

10.1.1. Cooperating Smart Everyday Objects

Siegemund identified several usage patterns of how smart everyday objects can cooperate with physically nearby smart objects and handheld devices in order to overcome resource limitations. He showed that computerized objects can make use of nearby devices (1) as mobile infrastructure access points, (2) as user interfaces, (3) as remote sensors, (4) as mobile storage media, (5) as remote resource providers, and (6) as user identifiers [Sie04b]. Siegemund further provided a software framework for realizing cooperative context-aware services based on smart objects. It consists of (1) a description language for context-aware services, (2) a context recognition layer for smart objects, (3) an infrastructure layer for distributing data among cooperating artifacts, and (4) a communication layer that adapts networking structures to correspond to the real-world environment of smart objects.

10.1.2. Fault-Tolerance as an Open Challenge

By providing support for localized cooperation based on the volatile computing context of a device (or its user), a physical smart object infrastructure can be used as the foundation for developing adaptive, fault-tolerant applications that make use of localized ad hoc redundancy. Due to the abundance of computation in ubiquitous computing settings, cooperation enables smart objects to access even more resources than they actually require [Sie04b]. So far, the challenge of *fault tolerance* and *reliability* with regard to smart object infrastructures has not been addressed. According to Siegemund, the question of how the abundance of computing resources in environments populated by large amounts of smart objects can be exploited for the implementation of more reliable services remains an interesting research question that requires further work.

10.1.3. Exploiting Localized Ad Hoc Redundancy for Fault-Tolerant Operation

Siegemund has shown how cooperating smart everyday objects can locally exchange sensory data via a distributed tuple space and interact with more powerful handheld devices by exchanging pieces of executable code called *Smoblets* [Sie04a, SK04]. This allows mobile devices to (1) exploit the computational resources of nearby handhelds in order to facilitate collaborative context recognition and (2) enabling graphical user interaction with smart objects by outsourcing user interfaces to nearby handheld devices.

For these reasons, we consider smart objects a valuable source of localized ad hoc redundancy. This redundancy can be exploited to provide mobile devices with fault-tolerant data transfer and communication services. In the following, we describe the design and prototypical implementation of a fault-tolerant and user-centric service infrastructure we developed based on cooperating smart everyday objects.

10.1.4. Dependability Issues Related to Cooperating Smart Objects

A fault-tolerant user-centric service infrastructure based on smart everyday objects has to take into account the fundamental dependability challenges stemming from cooperation among dynamic and autonomous entities.

Volatility of Cooperative Relationships: Smart everyday object infrastructures are usually highly dynamic and unpredictable with regard to the local availability of resources. A smart-object-based infrastructure for the delivery of reliable services based on localized ad hoc redundancy (cf. Chapter 4.5.5) has to deal with the volatility of cooperative relationships (cf. Chapter 3.3.10). Smart objects encountered in the vicinity of a mobile device not only are liable to cease to cooperate momentarily without prior notice, but they may do so permanently. For instance, mobile devices in general are expected to encounter smart objects along the way which they will never meet and interact with again. Fault tolerance methods therefore have to consider this volatile nature of relationships and adapt their functionality accordingly.

Incentives for Ad Hoc Cooperation: In our world model, we assumed that smart objects cooperate voluntarily. For closed user groups and settings, this can be a valid assumption. In open real-world systems, however, one cannot take it for granted that smart everyday objects are willing to cooperate without receiving some sort of reward for doing so. An important challenge therefore is the creation of incentives for ad hoc cooperation among independent, anonymous entities.

10.1.5. Dependable User-Centric Data Dissemination

The dissemination of data originating from the user to be transmitted to remote addressees by harnessing localized ad hoc redundancy provided by proximate smart everyday objects constitutes a user-centric dependability challenge. Concretely, we are interested in how personal data collected on the user's individual device can be reliably transferred to a specific remote back-end server for performing an application- and user-specific processing. We therefore consider the data dissemination process to be *user-centric*. Data dissemination further is a one-way data transfer: once some particular user data has been transferred, it is no longer of interest to the individual user (i.e., to the user's individual devices) and can be safely discarded.

The focus of a *dependable* system for user-centric data dissemination lies on the challenge of how individual users can exploit locally available resources in a self-centered manner to achieve the goal of a complete and timely data transmission to a remote communication partner.

This problem is inherently different from the *system-centric* challenge of ensuring the persistence and availability of data that is of interest for a larger user community, which is addressed by peer-to-peer data and file sharing systems, for instance. It is also different from the challenge of ubiquitous data access [ZHH03, HH04] where a user operates on personal data by means of different types of devices in different locations. Here the main research issues are automatic data selection, hoarding, and synchronization [HKZ02], and transcoding, with the goal of enabling continuous availability of data regardless of user mobility and temporary disconnection, and regardless of the individual mobile device itself and its data viewing/processing applications.

10.1.6. Employing Proximate Smart Objects as Local Communication Gateways

Smart everyday objects can serve as infrastructure access points [Sie04b] to mobile devices. In particular, cooperating smart objects that feature both short-range ad hoc and infrastructure-based communication capabilities can be employed by other mobile devices as *local communication gateways* that provide connectivity to remote services.

Mobile user devices that are capable of short-range ad hoc communication can exploit the communication capabilities of nearby smart objects in different ways: they can (1) *overcome temporary disconnections* and (2) *save energy* by using the communication capabilities of smart objects to connect to remote services.

10.2. Conceptual Framework

10.2.1. World Model

Our work is based on the following *world model*. A user carries with him a physical mobile or wearable user device, which we simply refer to as *mobile device* (MoD) in the following. A MoD features short-range ad hoc and infrastructure-based communication capabilities. Each MoD further has a digital representation in the form of a *virtual counterpart* [RSMD04]. The virtual counterpart (VC) of a MoD permanently resides in the background network infrastructure – typically a local area network (LAN) or the Internet – under a well-known address and with a unique identifier. The VC not only serves as communication proxy, but also features memory storage, processing, and communication capabilities that enable it to pro-actively perform management-related and application-specific tasks. Besides, the user's environment is populated with diverse physical computerized objects and artifacts, which we call *smart everyday objects*, or, in short, smart objects. A *smart object* (SO) features processing, data storage, and ad hoc communication capabilities. Each smart object has a unique identifier and address. In the case of *mobile* smart objects, remote interaction and communication is also handled by means of VCs residing in the background infrastructure. A smart object further can have some means of infrastructure-based communication. The smart objects are cooperative, which means they allow other smart objects or MoDs to make use of some of their memory storage and communication capabilities that are unused and redundant at the time of a resource sharing request. Concretely, smart objects support MoDs (1) in the *dissemination of data to remote addressees* (asynchronous *storing and forwarding* of data), and (2) in the *communication with remote entities* by acting as *local communication gateways*, with the goal of *sustaining local operability* of MoDs in the face of resource bottlenecks. As part of the infrastructure-based network infrastructure, *back-end* systems (BE) interact with the MoDs to provide user-specific applications and services. An overview of the world model and the involved entities is shown in Fig. 10.1.

10.2.2. Continuous User-Centric Data Dissemination

The dissemination of information to mobile users by delivering information in advance to physical locations where the user may need it (*hoarding*) has received considerable attention in mobile and ubiquitous computing research [TLAC95, KP97, KR99, HKZ02, VTB04]. In our system, however, the flow of information and data from remote sources to the user as the data sink only plays a minor role – this data flow is limited to the transmission of configuration information and short messages, for which a proxy-based approach is sufficient even under weak connectivity featuring a low bandwidth and a high delay. Instead, we investigate the *inverse* operation to hoarding: the user's MoD is the data source and user-specific information is to be transmitted to a specific remote data sink (i.e., the back-end system). The data transfer from a MoD to a back-end system can be performed directly using a direct communication link, or indirectly by using proximate smart objects as intermediaries (see arrows labeled "data" in Fig. 10.1). What is important from the perspective of the user is that his or her data arrives timely and completely at the designated addressee. The user is the origin of the data dissemination process,

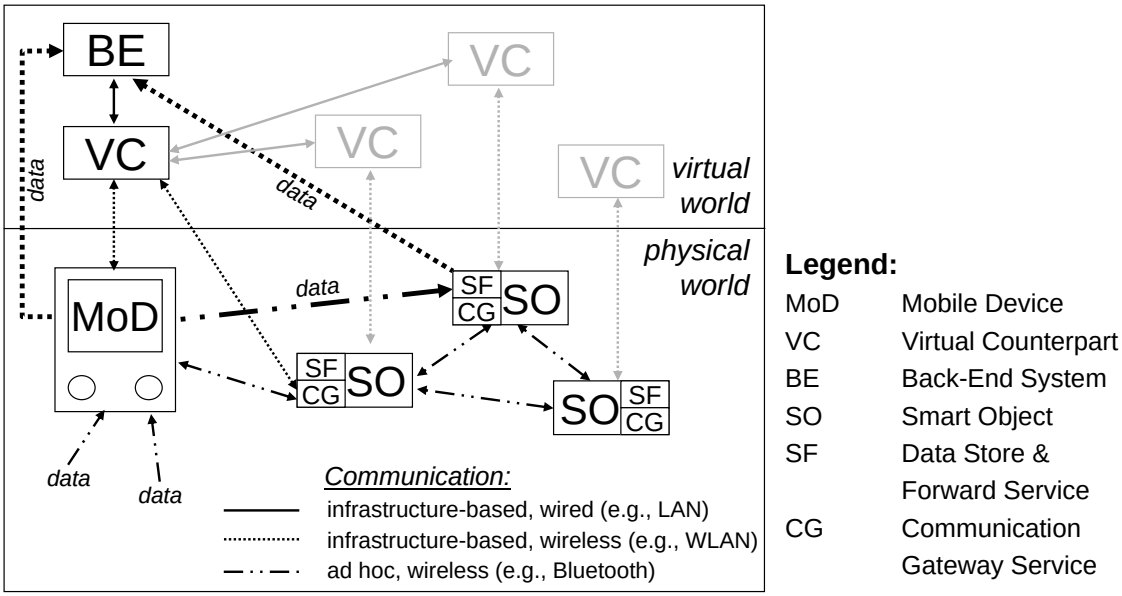

Figure 10.1.: World model of the fault-tolerant service infrastructure based on smart everyday objects

and the addressee is specifically related to the user in some way, which is why we call this type of data dissemination *user-centric*.

We further want to ensure that the data dissemination can be performed *continuously*, which is important in cases where the data collection on behalf of the MoD is a continuous process. For instance, continuous data collection and dissemination is typically of importance in *surveillance* applications, such as medical monitoring systems (e.g., measuring the heart activity of risk patients) or safety surveillance systems for small children and elderly people, for instance.

10.2.3. Role of Virtual Counterparts

In our system, the VCs residing in the infrastructure-based network perform the following functions:

Asynchronous Communication Proxy: A VC can act as an asynchronous communication proxy for remote, infrastructure-based communication. This enables other networked entities to communicate and interact with the MoD without knowing the current physical location or address of the MoD that is liable to change over time. The MoD regularly contacts its virtual counterpart upon connectivity in order to retrieve the latest messages or requests obtained from remote entities. For instance, a back-end system may send configuration updates to the VC, which the MoD then picks up eventually.

Persistent and Stateful Recovery Proxy: The VC also persistently maintains the internal state of the MoD, including user data, configurations, and state information about active cooperative relationships with other mobile or stationary entities. The VC may outlast the lifetime of an individual physical MoD. This enables the user to replace the MoD with a substitute (e.g., in

case of a defect or in case of loss) while preserving individual configurations and internal state information.

Fault-Tolerance Management: The VC is an active software component that manages various fault-tolerance aspects of its corresponding MoD. It can proactively interact with other entities, such as with the MoD, with back-end systems, or with the VCs of smart objects.

10.2.4. Generic Services of the Smart Object Infrastructure

In order to minimize the requirements with regard to the capabilities of smart everyday objects, we assume that a smart object provides one or both of the following services:

Asynchronous Data Storing and Forwarding: Based on their memory storage and infrastructure-based communication capabilities, smart objects provide the service of temporarily storing data from other smart objects or devices and eventually transmitting this data to a remote data sink. We refer to this service as *asynchronous store-and-forward* service. The point in time when the data is forwarded depends on the availability of connectivity to the back-end system. We assume that the smart objects in general do not behave maliciously or selfishly, but that they transmit any data that they accepted for storing and forwarding at best effort. This means that if connectivity is given and if the resources of the smart object are sufficient, the smart object will transfer the data as requested. However, as we consider the participating smart everyday objects to be independent and part of an open, fully decentralized ubiquitous computing environment, cooperation is voluntary and no quality-of-service guarantees are given. As a consequence, a smart object may discard previously accepted data without further notice in case of an imminent local resource bottleneck (e.g., memory shortage or low battery), or in case of a prolonged unavailability of the back-end system.

Synchronous Communication Gateway: By coupling ad hoc and infrastructure-based communication capabilities, smart objects can serve as local communication gateways. The corresponding service smart objects can offer to other smart objects and devices is the establishment of a synchronous communication link to destinations within the infrastructure-based network, which we refer to as *synchronous communication gateway* service. Potential clients connect to this service via short-range ad hoc communication. For instance, a MoD can indirectly connect to its VC or to a back-end system to exchange state information or messages by using the gateway service of a proximate smart object.

10.2.5. Inter-Relationships Between Entities

We assume that the user is carrying a mobile device that – at times – is liable to generate or collect a significant amount of data that has to be delivered *lossless* to a remote data sink (the *back-end*). For doing so, the MoD can directly connect to the remote back-end by means of wireless infrastructure-based communication if it

has the required local resources (energy, connectivity) at its disposal. However, in situations where no direct connectivity to the back-end is available, or when the energy level of the MoD is too low to maintain long-range wireless communication, the MoD can also interact in an ad hoc fashion with physically near smart everyday objects to make use of their data storage and communication capabilities. The idea is to enable the MoD to (1) save energy by using energy-efficient short-range communication, and to exploit the local data storage and connectivity of smart objects (SO) for (2) transferring data to the remote back-end (BE) or for (3) synchronizing the internal state of the MoD with its VC. In Fig. 10.1, the inter-relationships between the different entities are visualized by means of arrows.

10.3. Architecture of a Fault-Tolerant User-Centric Service Infrastructure Based on Cooperating Smart Everyday Objects

In the following, we outline the architecture of the fault-tolerant and user-centric service infrastructure based on smart everyday objects, which we developed and prototypically implemented (see Sect. 10.11).

10.3.1. System Architecture

We developed a layered system architecture for the smart-object-based data dissemination infrastructure. An overview of the system architecture is shown in Fig. 10.2.

10.3.2. Communication Layer

On the lowest level, in the *Communication Layer*, we find the basic communication mechanisms employed by smart objects and MoDs. In our system, we used Bluetooth [BS06] for short-range ad hoc communication to enable localized interaction, and Wireless LAN (IEEE 802.11b [WFA06]) for long-range infrastructure-assisted communication to connect to remote services provided as part of the background infrastructure (i.e., to connect to services offered by VCs or back-end systems).

10.3.3. Smart Object Layer

The *Smart Object Layer* provides a service interface for the interaction between smart objects. It implements the two generic services *asynchronous store-and-forward* and *synchronous communication gateway* described in Sect. 10.2.4. In addition, it provides additional management functionality for the provisioning, discovery, and the quality-of-service description of the services offered by individual smart objects. A smart object can possess multiple communication interfaces for long-range infrastructure-based communication (e.g., a stationary smart object can be linked to a LAN by wire and possess Wireless LAN or Bluetooth capabilities at the same time). In this case, the management functions can be used to parameterize the store-and-forward and communication gateway services during each individual use to employ a particular available communication interface.

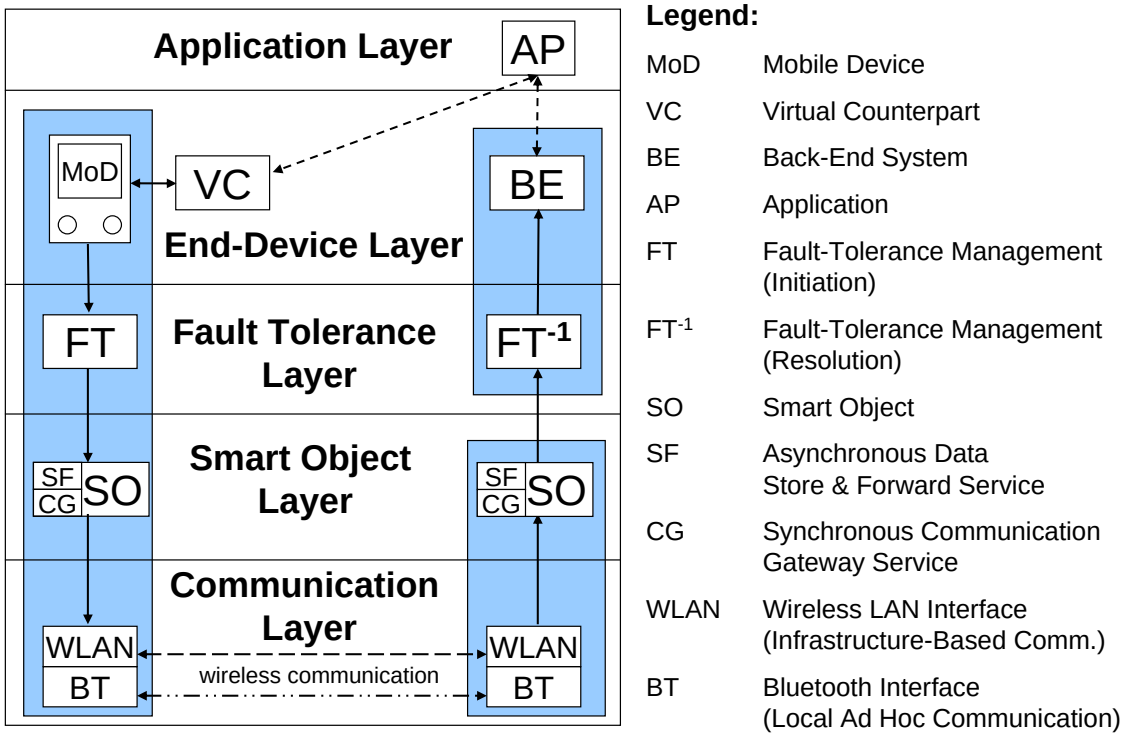

Figure 10.2.: Layered system architecture of the fault-tolerant service infrastructure based on smart everyday objects

Each MoD also implements the interface of the *Smart Object Layer* to be capable of interacting with smart objects (see Fig. 10.2). The MoD itself can also act as a smart object and share its resources and capabilities with other devices.

10.3.4. Fault-Tolerance Layer

The Fault-Tolerance Layer manages the fault-tolerance aspects of the data dissemination process and of the internal state of the MoD. The fault-tolerance procedures of the data dissemination are different with regard to the data producer or data source, which in our case is represented by the MoD and its VC, and the data receiver or data sink, which in our model is represented by a back-end system residing in the background infrastructure. The developed fault-tolerance mechanisms are described in Sect. 10.4.

10.3.5. End-Device Layer

The End-Device Layer provides an interface for sending and receiving data asynchronously (data dissemination), and for synchronous communication. It transparently employs the fault-tolerance mechanisms of the fault-tolerance layer, hiding the complexity of the fault-tolerance management from applications. We distinguish three types of end-devices: *mobile devices* (MoDs), their *virtual counterparts* that constitute the data sources, and *back-end* systems that represent the sinks of the data dissemination. Like the back-end systems, the VCs of MoDs also reside in the background infrastructure (network).

10.3.6. Application Layer

The application layer contains applications that interact with MoDs and back-end systems. Applications can retrieve information received by end-systems, or interact with MoDs by exchanging messages or changing device settings via the interface offered by the End-Device Layer. The interaction between an application and the MoD of a user is mediated by the corresponding VC, which both serves as proxy and intermediating representative. As an exemplary application, we describe an implementation of a user-centric and fault-tolerant patient monitoring system in Sect. 10.11.

10.4. Fault-Tolerance Management of the Fault-Tolerance Layer

10.4.1. Fault-Tolerance Manager Component

The functionality of the Fault-Tolerance Layer is managed by a *fault-tolerance manager* (FTM), which is an active software component. Each of the end-systems of the End-Device Layer (back-end, MoD, and VC) possess an FTM instance that performs particular fault-tolerance management tasks.

10.4.2. Fault-Tolerant Data Dissemination by the MoD

On the side of the *data source*, the FTM on the MoD (1) monitors the state of the device and detects critical resource conditions, (2) discovers available store-and-forward services offered proximate smart objects in their different forms, and (3) performs the selection and execution of one or more store-and-forward services depending on the current contextual situation. Step (3) involves the splitting of data into different payloads that are then disseminated in a redundant fashion by multiple smart objects.

10.4.3. Fault-Tolerance Management between MoD and Virtual Counterpart

The FTM of the MoD regularly contacts the FTM of the VC to perform the following operations: (1) retrieval of acknowledgments received from remote systems for previously disseminated data units, (2) synchronization of configuration data and application-specific settings that are mirrored by or were updated in the VC, and (3) reinitialization of the MoD with the configuration information stored in the VC in case the MoD loses its internal state due to an energy shortage, malfunction, or due to the physical replacement of the MoD with a substitute. For the interaction between the MoD and its virtual representation, we used a proprietary protocol we developed.

10.4.4. Data Reconstruction at the Back-End Server

On the side of the *data sink*, the FTM is responsible for the task of (1) putting together and recombining the redundant data transmissions received from different

smart objects. This is the complementary activity to the data splitting and redundant dissemination performed in step (3) at the data source (see tasks labeled with FT and FT^{-1} in Fig. 10.2). The fault-tolerance management in the data sink also (2) notifies the MoD about obtained or missing data packets by sending negative or positive acknowledgments to the VC of the MoD, which passes this information on to the device. Last but not least, the FTM (3) informs any smart object that tries to forward data that had already been received earlier at the sink from other smart objects that the data is no longer required. In this case, the concerned smart objects simply delete the obsolete data, and no duplicate data forwarding occurs.

10.4.5. Adaptive Selection of Communication Interfaces

The FTM further controls the selection and usage of available communication interfaces on the MoD by evaluating the current state of the MoD with regard to system resources. The FTM also respects semantic knowledge about the situation of the device (i.e., application-specific parameters that indicate an emergency situation/priority transmission). In the process, the FTM considers both communication interfaces for infrastructure-based communication on the device as well as on nearby smart objects.

10.4.6. Prevention of Imminent Failures

If localized ad hoc redundancy provided by a highly volatile smart everyday objects infrastructure is employed for the realization of a fault-tolerant data dissemination, it has to be ensured that the overall dependability of mobile applications does not suffer from the additional unpredictability and uncertainty introduced by the smart objects.

For achieving this goal, the FTM exploits the redundancy of the smart object infrastructure in situations where otherwise (1) data loss or failure of the MoD is imminent, or (2) where the successful completion of smart-object-based services can be validated.

Firstly, the FTM monitors the condition of vital resources of the MoD, which in our case are energy and memory space. If the amount of free memory space or the battery level drops below a critical threshold, the FTM tries to resolve the resource shortage by employing locally acquired ad hoc redundancy. Here the dependability of the system is not diminished by relying on volatile smart objects, as failure of the MoD would be imminent otherwise. Secondly, if memory space on the MoD is not critical, the services provided by a smart objects infrastructure can be used in a safe manner: by waiting for acknowledgments sent by the data sink before deleting data units on the MoD that were transmitted via store-and-forward services, the MoD can verify that these data units have been properly transferred and received at the remote system.

10.4.7. Proactive Situation-Aware Fault-Tolerance Management

We identified several fault-tolerance mechanisms that are performed according to the particular contextual situation of the MoD. The situation of the MoD is de-

termined by its internal resource availability (memory, energy), by the available external resources (services offered by proximate smart objects), and by semantic requirements of the application that affect the importance of certain tasks.

On the MoD, FTM continuously monitors the internal state and condition of the device. If the FTM detects a critical condition, it first establishes the computing context of the MoD (i.e., the available services offered by nearby smart objects). Then it applies a heuristic that selects and activates fault-tolerance mechanisms according to the established contextual situation of the MoD.

10.4.8. Semantic Prioritization of Fault-Tolerance Mechanisms

For rating the urgency of tasks that have to be performed by the MoD in case of a critical condition, we support *semantic prioritization*: (1) *low* priority for data that can be disbanded in case of resource shortages, (2) *normal* priority for data that are to be disseminated eventually without particular timing restrictions (default value), and (3) *high* priority for data that has to be disseminated as fast as possible or before a certain deadline. The priorities have to be set according to the requirements of a particular application – the prioritization therefore reflects a semantic application- or user-specific rating. During operation, these priorities are used by the FTM to establish the semantic situation of the MoD in case of a critical condition in order to select the appropriate fault-tolerance mechanism.

10.5. Internal Classification of Resource Conditions

To capture the condition of the resources of the MoD in a human-understandable way, we introduced a three-step state classification scheme, which we describe in the following.

10.5.1. Three-Step Resource State Classification (Green-Yellow-Red)

In our prototype system, we use a three color encoding to indicate the state of resource availability and the severity of adverse resource conditions:

Green: Indicates a *normal* resource condition and fault-free operation. The resource is available in ample, abundant quantity.

Yellow: Indicates an *abnormal* resource condition or operation which – if no actions are taken – can develop into a critical condition (e.g., low memory, no/weak communication, or energy shortage) that is liable to lead to the failure of the device. User intervention may be required/requested (e.g., replacement/recharging of battery; moving closer to nearby smart objects or communication hot-spots).

Red: Indicates a *critical* resource condition or faulty operation liable to cause the imminent failure of the data dissemination process or of the general operability of the MoD. The FTM issues a warning to the user according to the

configuration and capabilities of the MoD (e.g., playing an acoustic warning sound on a loud-speaker; displaying a visual warning message on the on-device display; sending a short message to the user's mobile phone, etc.). In addition, based on the particular applications executed on the mobile device, the MoD may also attempt to notify concerned back-end systems in high priority communication mode before the failure occurs.

10.5.2. Classification of Memory and Energy Resource Conditions

The states of the resources energy, memory space, and connectivity are rated according to the resource state categorization described in Sect. 10.5.1. For instance, with regard to *memory storage*, the different resource conditions are defined by two thresholds t_y^m and t_r^m: (1) condition *green* is active if the amount of available free memory space is plenty, which corresponds to a value above threshold t_y^m. (2) If the memory space drops below the value of t_y^m but remains above t_r^m, then state *yellow* is active, which means that the MoD is expected to run out of memory space soon. (3) If the amount of memory space drops below threshold t_r^m, condition *red* is active, which signifies that there is hardly any free memory space left and data loss is imminent. Likewise, the state of the *energy level* is determined by two threshold values t_y^e and t_r^e that define the energy levels for a state change from *green* to *yellow* and from *yellow* to *red*, respectively. The thresholds can be determined manually by the user, remotely by a third party, or dynamically by the applications on the MoD. The thresholds are set as percentage values describing the proportional quantity of a resource that has to be available. In the prototype system described below, we set the threshold values t_y^m and t_y^e for condition *yellow* to 20%, and the values t_r^m and t_r^e for condition *red* to 5% for describing the state of the resources memory and energy. That means that if the energy level drops below 5% of the overall capacity of the battery of the MoD, the condition of the energy supply is considered to be critical (condition *red*).

10.5.3. Classification of Connectivity Condition

The state of connectivity required for data dissemination and communication is determined by the type of available communication interfaces. Connectivity is in state *green* if the MoD itself or at least one proximate cooperating smart object features infrastructure-based connectivity providing asynchronous store-and-forwarding and synchronous communication gateway services. It is in state *yellow* if there is no direct connectivity to the background infrastructure but if there is at least one cooperating smart object nearby with store-and-forward capabilities (i.e., data can be transferred to the smart object and scheduled for later forwarding). In the worst case the MoD has no connectivity to the background infrastructure, and there is no smart object nearby that offers a store-and-forward service, which equals state *red*.

10.5.4. Intuitive Indication of Resource Conditions to the User

By representing states with colors in the style of a traffic light, the conditions of device resources can be indicated in an intuitive manner to the user. *Green* signifies that currently there is no problem with a resource, *yellow* indicates that a resource still is working fine but requires special attention as it may become critical any time soon, and *red* signifies that a resource has reached a critical condition that requires immediate intervention to prevent failure. The visualization of the resource conditions in such a color-encoding further enables the user to instantly and easily verify the effect of his or her corrective measures. For instance, after having moved near networked smart objects, the user can immediately verify the change of connectivity indicated by the color coding. And in the case of energy, the color coding allows to inform the user early that the replacing or recharging of the batteries of a MoD is recommended to be performed "soon" (e.g., charge overnight), showing resource state *yellow*, or "immediately" (e.g., replace batteries immediately), showing resource state *red*.

10.6. Fault-Tolerance Mechanisms Based on Proximate Smart Objects

The FTM supports a number of fault-tolerance mechanisms to prevent data loss or to maintain operability of the device, which we present in the following sections.

10.6.1. Preventing Data Loss by Exploiting Localized Ad Hoc Redundancy

If the MoD runs out of local memory space while it is collecting or generating data for dissemination to a remote back-end server, and if the MoD has no or insufficient connectivity, data loss is imminent. In that case, the FTM exploits localized ad hoc redundancy offered by proximate smart objects. More concretely, the FTM makes use of the data storing and forwarding capabilities of nearby smart objects. To increase the probability of successful (and timely) data dissemination, the FTM can charge multiple smart objects at the same time with the transmission of the same data load (payload), as individual smart objects – over which the MoD has no authority and control – may fail to complete the data transfer.

The number of smart objects that are employed for a single data load (degree of redundancy) is chosen according to an internal rule-based engine that takes into account the energy level of the MoD, the priority of the payload (normal or prioritized/emergency), the quality of service provided by the individual smart objects (i.e., free memory space, available communication capabilities for infrastructure-based communication together with their individual bandwidths and costs of usage).

Ideally, the sending of a single payload to several smart objects for dissemination should be performed via a *multicast* operation for maximizing the energy efficiency of the sender (i.e., the MoD), and for keeping the overhead low that increases with each additional smart object involved in the redundant data dissemination

procedure. In our case, the available short-range ad hoc communication technology (i.e., Bluetooth) did not support multicast operations for data dissemination but required a series of one-to-one interactions.

10.6.2. Maintaining Continuous Connectivity

The diverse infrastructure-based communication capabilities of smart everyday objects can be exploited by a MoD to maintain *continuous* connectivity at times when the MoD itself temporarily has no infrastructure-based connectivity.

In our system, if at a point in time the MoD itself lacks infrastructure-based connectivity but needs to disseminate user data to a remote back-end server at high priority (i.e., with stringent timing restrictions), the FTM tries to find one or more smart objects that feature active infrastructure-based connectivity. It then uses the store-and-forward capabilities of these smart objects and requests immediate transmission of the data.

The FTM also transparently employs the synchronous communication gateway services offered by smart objects to save the state of the MoD to the corresponding VC in case a critical condition has been detected (e.g., prior to an anticipated power outage), or to send urgent notifications to remote systems. For instance, a failing device could send a last short warning message to the user's mobile phone via an Internet gateway to prompt him to take corrective action.

10.6.3. Increasing Lifetime of MoDs Through Energy-Aware Communication

Energy generally is a limited resource of portable user devices. To increase battery lifetime and thus prolong the operability of a MoD, the FTM performs an *adaptive energy management*. If the FTM detects a low energy level on the MoD, then it dynamically switches from using the built-in long-distance communication capabilities to employing the communication capabilities of proximate smart cooperating objects. The idea is to save energy by using low-energy short-range ad hoc communication for the interaction with nearby smart objects in favor of direct long-range infrastructure-based communication that is more energy consuming.

In our case, using Bluetooth[1] for local ad hoc wireless communication instead of Wireless LAN[2] for long-distance infrastructure-based communication, energy consumption can be reduced by up to approx. 80%, for instance).

In our system, the low-power communication capabilities of smart objects are used in two ways by the FTM: (1) to disseminate data to a remote back-end via the store-and-forward services of nearby smart objects, and (2) to connect to the virtual counterpart or to other remote entities for message passing and data synchronization.

[1]Bluetooth Class 2 standard [BC06]; worst case peak current consumption of 60 mA at a transmitting power of max. 2.5 mW (4 dBm) and a range of up to 10 m.

[2]IEEE 802.11b standard [WFA06]; current consumption approx. 300 mA at approx. 100 mW (20 dBm) transmitting power and a range of approx. 100 m.

10.7. Rule-Based Activation of Fault-Tolerance Mechanisms

In our system, the behavior of the FTM is controlled by a rule-based engine, which uses a customizable set of rules that select and parameterize fault-tolerant mechanisms according to the situational context of the MoD (i.e., the state of individual resources and the priority-level of data that is to be disseminated). The syntax of the rules makes use of the classifications of resource conditions described earlier. This enabled the creation of human-readable and intuitively editable sets of rules for individual fault-tolerance mechanisms.

Tables 10.1 and 10.2 show exemplary rules in tabular notation as they are used in our prototype system described in Sect. 10.11.

10.7.1. Selection of Direct and Indirect Data Dissemination

Table 10.1 declares when the FTM has to use the built-in infrastructure-based communication capabilities of the MoD (entry *true*) and when it should rely on the store-and-forward services offered by nearby smart objects if available (entry *false*). The rules consider the state of the energy supply and of the available memory space of the MoD, and the prioritization of the data to be disseminated.

For instance, we can see in the table that for *high priority* data, the FTM always tries to disseminate the data directly via the built-in infrastructure-based communication interface if the energy level is not yet critical (condition *red*) or if memory space on the MoD is critical. The asterisk symbols after the true/false entries indicate that the alternative means of data transfer is selected if the specified data transfer mode is not available. That signifies that if no direct connection to the back-end is available due to insufficient connectivity, the FTM will always try to use nearby smart objects instead. Likewise, if no smart objects are available, direct data transfer is used if connectivity is available.

In the case of data with *normal priority*, direct data transfer is only allowed if the energy level is ample (condition *green*) and the free memory space is abnormal or critical (condition *yellow* or *red*, respectively). However, the asterisks indicate that if no direct data transfer is possible, indirect data transfer is to be used instead. As long as the internal memory of the MoD is in normal condition (*green*), only data dissemination mediated by proximate smart objects is allowed. If no smart objects are available, no data transfer occurs until the resource conditions change and a new rule applies that may enable direct data transmission. For instance, if memory is in state *red* and energy in state *yellow*, the use of the direct data transfer capabilities of the MoD is allowed if no indirect store-and-forward services are available.

For data with *low priority*, the FTM always requires *indirect* communication and data dissemination via smart objects (table not shown). This means that the FTM accepts the possibility of data loss. For instance, data loss may occur in situations where the unavailability of smart objects leads to a critical memory condition on the MoD.

Normal	*Memory*		
Energy	**green**	**yellow**	**red**
green	false	true*	true*
yellow	false	false	false*
red	false	false	false

High	*Memory*		
Energy	**green**	**yellow**	**red**
green	true*	true*	true*
yellow	true*	true*	true*
red	false*	false*	true*

Table 10.1.: Two rule sets in tabular notation determining the use of direct or indirect data dissemination for normal (left table) and high priority data (right table). *True* signifies the selection of built-in communication capabilities, and *false* the use of smart objects. Settings marked with an asterisk are inverted (*true*→*false* or *false*→*true*) in case the resources needed for fulfilling the particular rule are unavailable

10.7.2. Determining the Degree of Redundancy of Indirect Data Dissemination

The rules in Table 10.2 determine the degree of redundancy (redundancy factor) that is to be applied by the FTM of a MoD when using indirect data dissemination based on the store-and-forward services of nearby smart objects. The figures in the table indicate the number of redundant store-and-forward services that the FTM should employ simultaneously for disseminating one and the same data payload. The decision is again based on the current internal memory and energy state of the MoD and on the semantic prioritization of the user data.

For *normal priority* data, the redundancy factor is zero if the energy level of the MoD is abnormal or critical (condition *yellow* and *red*, respectively) while free memory is abundant (condition *green*). This means that in these situations, no data is outsourced to smart objects until the energy level is back to normal (condition *green*). In all other situations, the redundancy factor is set to a factor $R \geq 1$, which means that the store-and-forward services of R different smart objects are used at the same time for disseminating a particular user data payload.

With regard to *high priority* data, all redundancy factors have been set to values ≥ 1. This ensures that data is to be disseminated by at least one smart object in all situations, even if the energy level of the MoD is critical. Further, one can see that the individual redundancy factors are higher than the ones chosen under similar resource conditions in the case of data with *normal priority*.

For data with *low priority* (table not shown), the redundancy factors are set to 1 for all rules that apply to normal energy level (condition *green*), and zero otherwise. That means low priority data is only disseminated when the energy level on the MoD is normal.

For the selection of smart objects out of a potentially large pool of candidates, the FTM further uses a heuristic that takes further parameters into account, such as the costs of individual services (see Sect. 10.8) and the amount of free memory space on the proximate smart objects. Detailed information about the applied heuristic can be found in [Geg04].

Normal	*Memory*		
Energy	**green**	**yellow**	**red**
green	1	2	3
yellow	0	2	2
red	0	1	1

High	*Memory*		
Energy	**green**	**yellow**	**red**
green	3	4	5
yellow	1	3	3
red	1	1	2

Table 10.2.: Two rule sets in tabular notation for defining the degree of redundancy applied to the smart-object-assisted dissemination of normal (left table, normal priority) or prioritized user data (right table, high priority)

10.7.3. Definition and Maintenance of Rules

The rules for selecting and activating fault-tolerance mechanisms have to be defined for individual MoDs (and their applications) by the respective users or by a system administrator. In our system, we provide a table skeleton for each rule that corresponds to the tables shown above. Further, the VC of an MoD maintains a copy of the set of rules. The VC further supports the remote update of individual rules via remote procedure calls. Changes to rules in the VC are updated on the MoD whenever the latter connects to its VC for data and state synchronization. For the synchronization procedure, we use a proprietary protocol we developed.

10.8. Incentives for Cooperation Among Independent Smart Objects

For the selection of services offered by smart objects for redundant indirect data dissemination we also considered *cost* as a factor (see Sect. 10.7.2). This is based on the assumption that cooperation among autonomous mobile devices that do not belong to the same authority (i.e., each MoD is its own authority) requires some sort of rewarding/payment scheme as an incentive. In our above-mentioned heuristic for selecting a number of smart objects out of a large pool of available candidates, we therefore included the cost of services provided by smart objects as a parameter, assuming the availability of a distributed payment/reward mechanism.

10.8.1. Stimulating Cooperation in Mobile Ad Hoc Networks

The problem of stimulating cooperation in mobile ad hoc networks is a fundamental challenge: the participating entities (also called *nodes*) are expected to cooperate in order to support the basic functions of the network. A major issue is that the individual nodes are assumed to be *selfish*, meaning that they try to maximize the benefits that they get from the network while minimizing their contribution to it.

The usage of payment schemes as an incentive for the fostering of cooperation among autonomous entities in open ad hoc networks has been investigated in [BH00, BH01, BH03]. The work focuses on a particular instance of the problem of providing incentives for cooperation: packet forwarding (routing). A virtual currency called *nuggets* is suggested as a means of payment among ad hoc cooperating nodes. Physical nodes in the ad hoc network are equipped with a stock of nuggets. Each node then "pays" using the nuggets if it originates a data forwarding request.

Two payment models were presented: (1) the Packet Purse Model loads the nuggets onto the packet and forwards it. The intermediate nodes acquire nuggets from this packet and then forward the packet. This approach has the disadvantage that the number of nuggets needed to reach the destination node is not accurately known in advance. (2) The Packet Trade Model does not load nuggets onto the forwarded packet, but instead "buys" each packet using nuggets and "sells" it to other nodes acquiring nuggets. A drawback of this approach is that billing a node does not discourage senders from flooding packets and overloading the network since they do not have to pay, thus abusing the ad hoc network consisting of cooperating autonomous nodes.

The problem of stimulating packet forwarding in self-organizing, mobile ad hoc networks for civilian applications based on cooperation among independent nodes has also been addressed in [BH03]. The described approach is using a counter-based mechanism and trusted, tamper-proof hardware for security protection.

10.8.2. (No) Need for Incentive Systems

Referring to the cooperative relaying of messages in open ad hoc networks consisting of heterogeneous, user-controlled devices, Huang et al. argue that in practice, the need of incentive systems is likely to disappear in the long run [HCW04]. They point out that in the case of mobile ad hoc devices, practically the only benefit to consumers who would not be willing to cooperate would be longer battery lives. As a consequence, for most users the gain of not participating or even of cheating is not worth the trouble of modifying their devices or tampering with the software. They also state that the addition of "excessive complexity" introduced by incentive systems may "hurt the deployment of ad hoc networks" in general, especially in the early stages of adoption. Besides, the more devices that do not cooperate (here: that do not relay messages), the worse the overall performance of the cooperative system (here: ad hoc network) will be, until it is finally completely useless. It is conceivable that similar considerations also apply in parts to service infrastructures based on cooperating smart everyday objects. The problem of stimulating cooperation and the question of the need of incentives, however, is beyond the scope of this work and requires further future research.

10.9. Support for Disconnected Operation

Our system architecture supports disconnected operation of the MoD with the assistance of (1) a VC residing in the background infrastructure, and by means of (2) localized interaction with smart everyday objects.

10.9.1. Maintaining Remote Relationships by Means of Virtual Counterparts

The VC remains active and reachable even if the physical MoD is temporarily disconnected from the background network infrastructure. This enables the MoD to maintain relationships with remote entities even while it has no infrastructure-based connectivity.

Data Dissemination Management: The FTM component on the VC manages acknowledgments received from back-end servers as part of the redundant, smart-object-mediated data dissemination initiated by the FTM on the MoD. Whenever the MoD reconnects to its virtual representation, the VC informs the FTM on the MoD up to which sequence number the data units disseminated by the MoD were already received and can be discarded, and which individual units are still missing and subject to retransmission.

Message Relaying: The VC accepts messages delivered from remote entities and passes them on to the MoD upon reconnection. On the MoD, the messages are delivered to the respective addressees according to the message type specified in the header. In our system, messages can be addressed to the FTM or to applications running on the MoD, containing configuration updates (e.g., new rules for the activation of fault-tolerance mechanisms as described in Sect. 10.7) or application-specific instructions, for instance.

10.9.2. Localized Ad Hoc Cooperation

In disconnected state, the MoD uses the memory storage capabilities of proximate smart objects when outsourcing data for dissemination to the back-end server as part of its fault-tolerance management. That means that the data dissemination procedure can be sustained as part of disconnected operation if a time-delayed data delivery at the back-end server is acceptable.

10.10. Further Dependability Issues

There are a number of further dependability issues related to our system, which were not in the focus of our work, but which we wish to address briefly in this section for the sake of completeness.

10.10.1. Data Confidentiality and Integrity

The confidentiality of user data that is outsourced to smart objects for forwarding can be secured by means of encryption schemes. The MoD can negotiate a key with the back-end server to be used for the encryption of user data during the data dissemination process.

The integrity of individual data payloads that are received at the back-end system can be protected and verified using CRC checksums, for instance.

10.10.2. Access Control

In an ideal open system environment, the usage of smart objects is unrestricted to MoDs. Each MoD itself in return should also constitute a smart object and thus participate in the ad hoc resource sharing and localized cooperation. In our system, we therefore did not consider issues related to access control.

There are situations, however, where the enforcement of access control policies could be required. For instance, the use of incentive systems may require the implementation of access control mechanisms. Furthermore, the protection against

denial-of-service attacks may require some form of (temporary) access control to prevent continuing abuse on behalf of rogue MoDs or misbehaving smart objects.

10.10.3. Device State Recovery and Life Cycle Management Support

The VC is capable of persistently maintaining the states of the MoD, of its FTM, and of the applications running on the MoD. This is achieved by means of a periodic state synchronization with the physical MoD whenever the latter reconnect to its VC.

If the MoD loses its internal state due to a failure, it can (partially) recover that state upon reconnection to its virtual counterpart. If the physical MoD is defect and has to be replaced by a substitute device of the same model, the state information maintained at the VC can also be used to initialize and bootstrap the replacement MoD. Thus the VC concept facilitates the life cycle management of mobile devices.

10.11. Prototype Implementation: A Dependable and User-Centric Mobile Patient Monitoring Platform

Based on the system architecture and fault-tolerance management issues described earlier, we prototypically implemented a system that provides fault-tolerant user-centric data dissemination and communication services to mobile user devices. In the following, we give an overview of the implemented services and applications that in their entirety form a rudimentary *mobile patient monitoring platform*.

For a more detailed description of the patient monitoring application, including an in-depth description of the implementation of the Fault-Tolerance Layer and of the Smart Objects Layer, please refer to [Maz03] and [Geg04].

10.11.1. Application Scenarios

As an exemplary application scenario, we chose a mobile patient monitoring application, because this type of surveillance scenario (1) usually benefits from continuous data dissemination, (2) involves small, resource-limited hardware devices that the user carries on his body, and (3) has strong dependability requirements.

Besides the patient monitoring scenario, there are many other ubiquitous computing applications conceivable where an infrastructure for dependable continuous data dissemination would be helpful. For instance, other potential applications are (1) a *smart miniature camera* which constantly transfers its snapshots to a photo collection maintained at a remote back-end server, (2) an *electronic diary* that continuously stores information received from other smart objects during the day or diary entries written by the user himself, or (3) a *digital pen* that instantly digitizes everything that is written [Mat01b] and sends the data immediately to a remote database server.

10.11.2. Exemplary Application Scenario: Ambulant Patient Monitoring

The intention of the mobile patient monitoring system is to make it possible for risk patients to be treated ambulatorily rather than stationary in the hospital, which enables the individual risk patient to lead a more normal life and supposedly increases his or her quality of life.

Concretely, we assume a risk patient who is equipped with a wearable device (i.e., the MoD) that continuously collects sensory data about vital aspects of the patient's health. This data is continuously transmitted to a back-end server in a hospital where the data is automatically evaluated to detect critical health conditions that require instant medical assistance.

In response to the monitored health condition of the patient, the doctor from time to time adjusts the medication for the patient – this information is then transmitted to the MoD, where it is displayed to the user. The doctor can also (re-)adjust thresholds that indicate an emergency situation with regard to the measured values of certain health properties.

In our case, for simplification, the MoD measures the patient's heart rate, the value of which is supposed to remain within a certain interval. If the MoD detects that the heart rate exceeds a lower or a higher bound, then it assumes the occurrence of an emergency situation. As a result, the MoD displays a warning message to the user, informing him to seek immediate medical treatment, and it attempts to send an emergency message (high priority) to the back-end server to call surveillance personnel for help because of a supposedly imminent health threat.

For the detection of unobtrusive health problems requiring a more complex processing at the hospital, and for the seamless study of long-term effects of the patient's health development, the monitored data has to be transmitted completely and in a timely manner.

10.11.3. Mobile Patient Monitoring Device

As a mobile patient monitoring device (MoD) we used an HP iPAQ handheld device of the H5450 series with the PocketPC 2002 operating system. The handheld device featured Bluetooth and Wireless LAN (IEEE 802.11b) connectivity. The software on the MoD was implemented using Visual Embedded C++ Version 3.0.

Figure 10.3 shows the graphical user interface (GUI) of the implemented patient monitoring application executed on the MoD. The GUI contains the following visual elements: an LED for the condition of the energy level (current setting: *green*) and one for the memory condition (current setting: *yellow*). To the right of the LEDs, the current communication mode is displayed underneath the label "Connection" (current value: BT = Bluetooth). On the far right, the current prioritization of the data is shown (current value: low). In the status window below, the current fault-tolerance-management activity of the MoD is logged and displayed. The condition of the energy and memory resources can be manually changed by way of the modifier-buttons next to the corresponding status LEDs, which enables the simulation of different resource conditions. Whenever the resource conditions are changed, the triggered fault-tolerance activities are displayed in the status window. For instance, in the figure, the list contains entries about (1) the sending of high

priority data, (2) several ad hoc data dissemination attempts where the store-and-forward services of three smart objects where requested but only one was available, and (3) an entry indicating that the configuration of the MoD was updated by using the communication gateway service of a smart entity through a short-range Bluetooth connection. Below the status window, the current MoD configuration with regard to the acceptable range of the user's pulse rate are displayed: the user's pulse is considered to be normal when in the range of 40–120. Underneath the currently advised medication (type and dose) is displayed, and the update frequency (in ms) that determines how often the MoD reconnects to its VC for retrieving configuration updates and for synchronizing its internal state. On the bottom right, the priority level of the data is displayed, and the current average heart rate of the user (here: 77 heartbeats per minute). The average pulse value can be edited to simulate situations of normal or critical heart rates. Currently, the mobile patient monitoring application is not connected to a real heart rate sensor, but it creates random heart rate values around the specified average value.

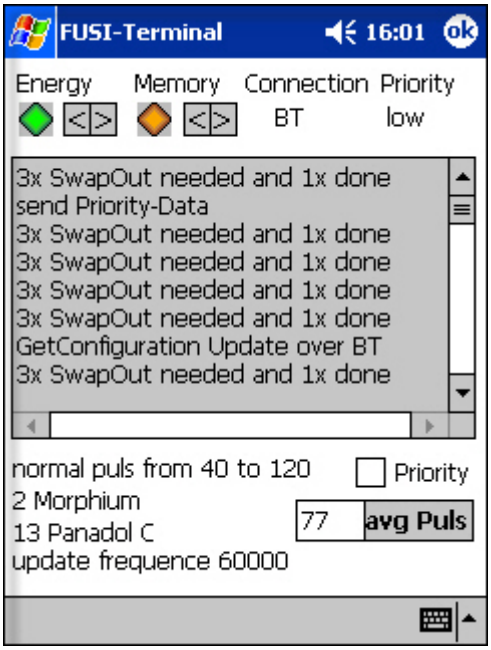

Figure 10.3.: Graphical user interface of the mobile patient monitoring device (MoD)

10.11.4. Back-End Server Application

The back-end server is represented by a Java server application residing in the LAN at a well-known address. The back-end server contains the logics for fault tolerance and data management. It also contains methods for receiving data transmissions from smart objects or MoDs, and for sending configuration updates or messages to a user's MoD. Both incoming and outgoing communication is socket-based (TCP/IP). Applications can register themselves at the back-end server for certain data units, based on the ID of the sender and type of data – the required information is specified in the headers of data units sent by MoDs.

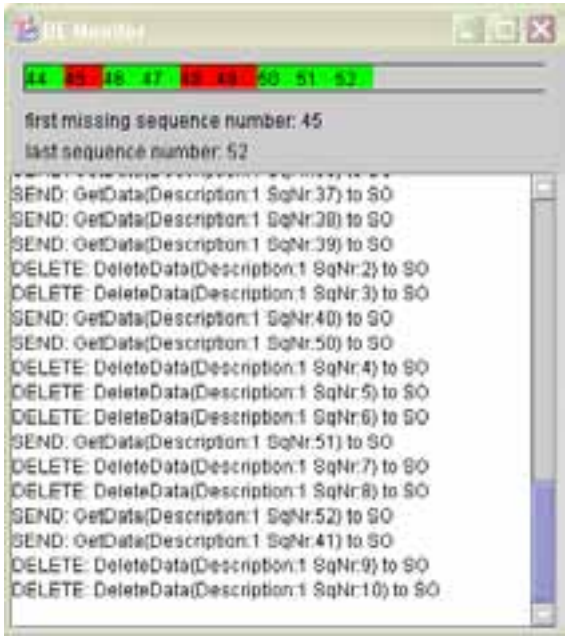

Figure 10.4.: Monitoring console of the back-end server application

The internal state and activities of the back-end server application during operation are logged by a graphical monitor application (see Fig. 10.4). On the top of the GUI, a status bar indicates which user data units have already been received, and which are still expected. For instance, in the figure, all data units with a sequence number $\leq$44 and with sequence numbers 46, 47, 50, 51, and 52 have been received correctly, while the data units with sequence numbers 45, 48, and 49 are still missing. Below the status bar, a status window is displayed which lists the latest operations executed on the back-end server (method *GetData* for receiving data units, and method *DeleteData* for informing smart objects about data units which can be safely deleted).

10.11.5. Infrastructure-Based Patient Surveillance Application

The back-end server receives and stores the user data obtained from associated MoDs. The *patient surveillance application* performs a continuous monitoring of the patient's health condition by interpreting that user data. It enables the medical personnel at the hospital, who are in charge of looking after the health conditions of the risk patients equipped with the mobile health monitoring equipment (i.e., the MoDs), to perform a remote diagnosis in case of an emergency, or to readjust the doses of prescribed medication.

For that, we developed and prototypically implemented a simple exemplary patient monitoring application. Its graphical user interface (GUI) is shown in Fig. 10.5. The top area of the GUI contains editable fields for setting the doses of the user's currently prescribed medication (Panadol and Morphium in our mock-up application). On the top right, the status of the user's current heart rate as well as the resource status of the user's MoD are displayed. Free memory space on the MoD is ample (condition *green*), but the mobile device is about to run out of energy

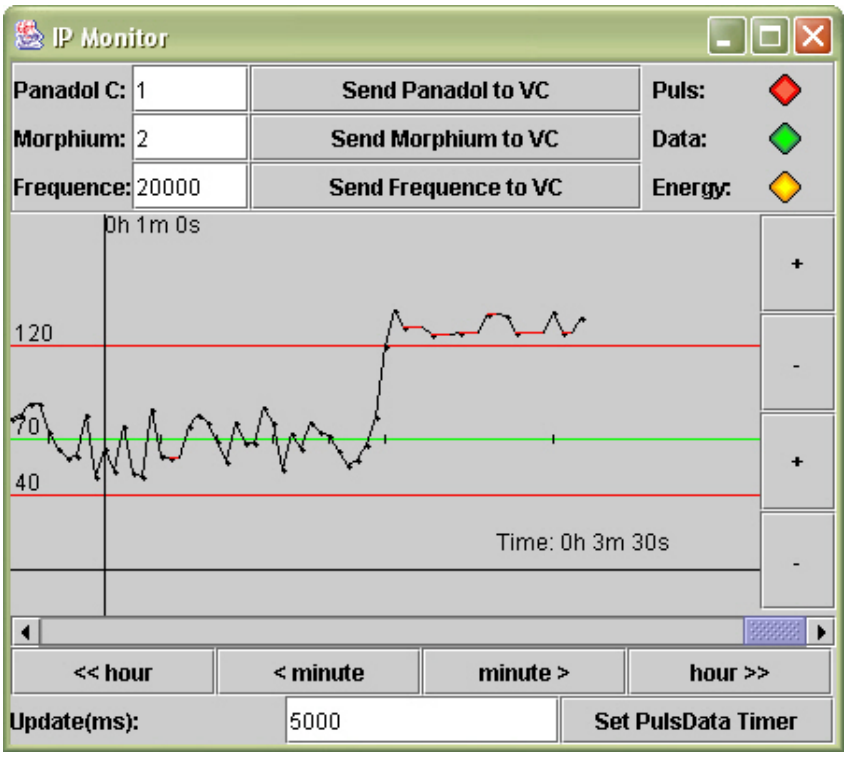

Figure 10.5.: Graphical user interface of the patient surveillance application

(condition *yellow*). In the middle area of the GUI, a visual graph illustrates the long-term development of the user's heart rate. The panel features an upper and lower threshold defining the interval of what is considered an uncritical heart rate for the particular user. Each threshold can be raised or lowered by clicking onto the two modifier-buttons on the far right. By using the buttons underneath the display showing the graph, the user can scroll backward and forward in time (by hours or minutes) to review the history of recorded heart rates and emergencies. On the bottom of the GUI, the update frequency for the displayed heart rate graph can be set.

In the figure, the heart rate is currently above the effective upper threshold, which means that the patient's current health condition is considered critical. This is also indicated by a *red* LED for the pulse status in the top right of the GUI.

10.11.6. Virtual Counterpart

We implemented VCs as stand-alone Java applications that reside in the LAN at well-defined addresses, and which communicate with other entities via TCP/IP connections. The public interface of a VC provides two main methods: one for (1) receiving remote messages from applications and from the back-end server, and another for (2) receiving incoming synchronization or state recovery requests from the MoD. For visualization and demonstration purposes, we also programmed a graphical user interface. It logs and displays the current internal state and recent activities of a VC (see Fig. 10.6). In the figure, the log indicates that configuration updates (obtained from the patient surveillance application) were forwarded to the MoD. Further, the log contains entries that indicate that the state of the MoD has been periodically synchronized with and saved by the VC.

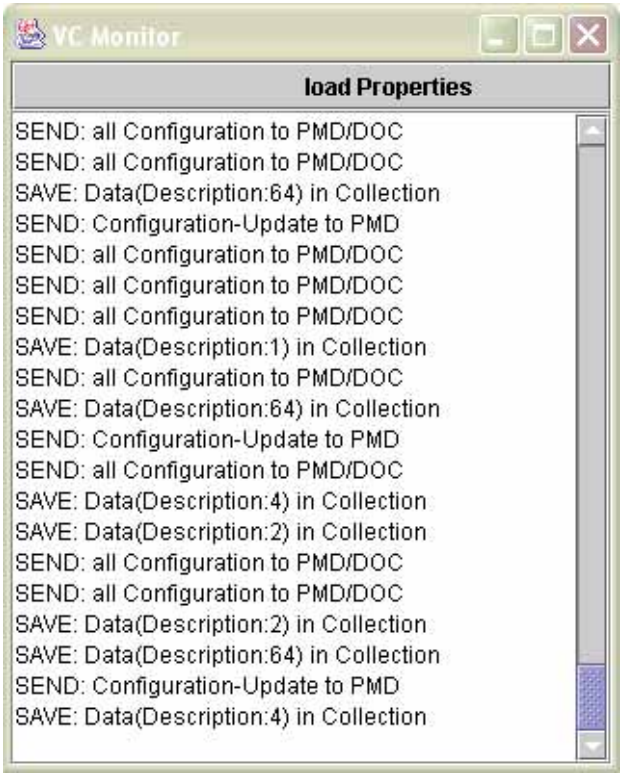

Figure 10.6.: Visualization of virtual counterpart activity

10.11.7. Smart Objects

In our prototype implementation, we emulated physical smart objects with networked Java applications. This allowed us to set up and execute an arbitrary number of smart objects for demonstration purposes. The smart objects were executed on different notebook computers featuring both infrastructure-based Wireless LAN as well as ad hoc Bluetooth connectivity.

In order to visualize the activities and to modify the states of individual smart objects, we implemented a graphical smart object monitoring application which is shown in Fig. 10.7. In the top left corner of the corresponding GUI, the current connectivity status of the smart object is displayed for the Wireless LAN (condition *red* = no connectivity) and for the Bluetooth (condition *green* = connectivity) communication interfaces. The connectivity status for the different communication interfaces can be changed by clicking on the LED. Further to the right, two editable fields enable the user to reset the amount of total and free memory that is available on the smart object. In the top right corner, the currently effective amount of total and free memory (which is subject to change over time as the smart object receives and forwards data obtained from MoDs) is shown. The recently performed activities of the smart object are logged in a status window below. In the current situation displayed in the figure, the log shows recent data store (STORE) and delete (DELETE) operations that were executed by the smart object as a result of completed store-and-forward requests. The log also shows responses to inquiries sent by the MoD about the free memory capabilities of the smart object.

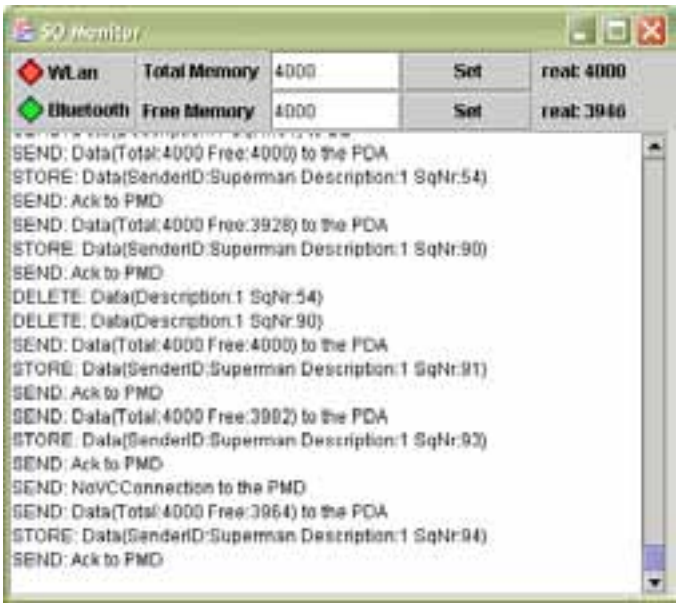

Figure 10.7.: Visualization of smart object state and activity

10.12. Conclusion

In this chapter, we presented a system architecture that achieves fault-tolerant operation by means of localized cooperation and resource sharing. The infrastructure exploits volatile user-centric redundancy provided by cooperative smart everyday objects, which form part of the local volatile computing context of mobile devices in ubiquitous computing environments (see Sect. 9.2.1).

Concretely, our system enables mobile, resource-limited devices to harness the redundant memory storage and communication capabilities of proximate smart everyday objects for achieving fault-tolerant data dissemination and communication, and energy-aware operation. The system puts mobile, resource-limited devices into a position to cope with faults stemming from the temporary unavailability of infrastructure-based connectivity, as well as with faults caused by resource bottlenecks concerning memory space and energy. By means of using a virtual counterpart as recovery proxy, the system also assists the user in recovering previously backuped settings and configurations in case the personal user device suffers from a failure that results in the loss of the internal state of the device. Finally, we presented a prototypical implementation of an exemplary patient surveillance application, which makes use of our fault-tolerant service architecture based on cooperating smart objects.

A limitation of the current version of our prototype system is that the smart object infrastructure is mainly represented by software processes running on desktop or notebook computers. Further work is required for developing and implementing a general smart object interface for a broader range of mobile and stationary devices as they appear in ubiquitous computing landscapes. In particular, smart object functionality should be supported by mobile phones, smart phones, and personal digital assistants, which already today constitute truly ubiquitous technologies (see Sect. 3.3.3). A further limitation is the local ad hoc interaction based on Bluetooth technology, which does not scale well to a large number of devices, takes long to complete, and consumes much energy during device and service discovery [SR03].

As part of future work, alternative ad hoc communication technologies such as ZigBee [Zig06], for instance, should be evaluated.

10.13. Related Work

10.13.1. Smart Objects vs. Super-Distributed Objects

Arbanowski et al. [ASRPZ03] presented a distributed computing environment that controls and manages large numbers of autonomous and decentralized objects, which they refer to as *Super Distributed Objects* (SDOs). The goal is to create a seamless, human-centered computing environment on the basis of users' individual communication spaces, which provides *I-centric* services that can adapt to users' individual requirements in individual living or working spaces. The notion of "I-centric" computing is used to emphasize that the system is centered on human behavior, which in the context of this dissertation corresponds to the notion of "user-centric" computing.

Super Distributed Objects [OMG04] can be any real physical entity or device, software component, or service. They are able to manage ad hoc communication between highly dynamic distributed objects. In the process, each individual SDO behaves autonomously, providing all necessary functions to enable an ad hoc service usage, and it continually interacts with diverse and interconnected objects while keeping the technology invisible to users. SDOs can further form dynamic groups (clusters) that provide services in a flexible and decentralized way in heterogeneous computing environments. To control all the heterogeneous entities in a common way, SDOs feature a well-defined open service interface, including various management interfaces for SDO service discovery, state monitoring, configuration, and resource reservation.

The so-called "I-centric Communications" approach based on SDOs enables a flexible ad hoc composition and/or configuration of services according to personal user preferences, which may change over time: it supports *adaptability* as the functionality of I-centric services is constantly adapted according to personal user preferences (*personalization*) and acquired knowledge about the environment (*ambient-awareness*).

Compared to the smart objects approach by Siegemund [Sie04b], which is concerned with the localized interaction and cooperation between autonomous *physical* entities, the "key factor for I-centric Communications" is considered to be service *personalization* and less the interaction between individual objects. In the I-centric Communications approach, applications are envisaged that allow individuals to assemble their service by simple "drag and drop" mechanism: similar to a LEGO toolbox, the individual should be able to create and to deploy its I-centric services [PZSA04]. Furthermore, to this time, *fault-tolerance* aspects of I-centric services have not been in the focus of SDO research.

10.13.2. Outsourcing of Code and Remapping of Applications

Marculescu et al. [MZSMM03] suggested techniques for the robust execution of applications in distributed, embedded failure prone environments such as found in

ubiquitous computing systems: the underlying execution environment consists of ultra low power, failure-prone wired-network sensors with low-end computing capabilities and possibly faulty communication and computation. The main idea is to enable applications to make use of the redundancy provided by "hundreds of computational devices" found in networked ubiquitous computing systems for enabling fault-tolerant operation. The proposed techniques are based on the concept of *code migration*: failing applications are remapped onto other hardware in environments with redundantly deployed hardware. The goal is to achieve (1) adaptability of applications to changes in the operating conditions (e.g., energy, memory, failures, etc.) and (2) graceful quality-of-service degradation in order to increase the lifetime of applications under various types of faults despite scarce energy resources.

The concept of application mapping and remapping could also facilitate the development of fault-tolerant mobile applications by exploiting localized ad hoc redundancy provided by smart objects. The *Smoblets* approach [Sie04a, SK04] described in Sect. 10.1.3 also enables the outsourcing of executable program code fragments to proximate physical devices for remote execution. Instead of relying on a wired sensor network, it uses a distributed tuple space that provides a localized shared memory space. The Smoblets approach could be modified to provide fault-tolerant local processing, data management, and the sharing of communication capabilities. With regard to the user-centric data dissemination, it is conceivable to upload executable program code onto nearby devices for enhancing a smart object in an ad hoc fashion with the asynchronous store-and-forward capabilities.

However, the execution of program code obtained from potentially untrusted third parties constitutes a major security risk with regard to malicious code. For reasons of trust and security, we do not consider the remapping of application code onto independent, autonomous devices a viable solution for achieving fault tolerance in *open* ubiquitous computing environments.

10.13.3. Heterogeneous Wireless Networks

Related to the idea of employing diverse communication capabilities is the work in the domain of *heterogeneous wireless networks*, which investigates means for mobile devices of using a variety of heterogeneous wireless access technologies. The goal is to provide support to mobile devices for performing seamless mobility, adaptivity, bandwidth aggregation, and end-to-end communication [Bha97, HKS04] over commercially available wireless networks. Further, heterogeneous wireless network architectures have been proposed that integrate cellular networks with wireless ad hoc networks [BWLW04, WH04]. Here major research challenges are the development of dynamic and adaptive routing protocols. In contrast to heterogeneous wireless networks, our approach abstracts from low-level communication issues such as routing and seamless hand-offs. Instead, our work concentrates on the creation of an application-oriented middleware architecture that employs redundancy provided by proximate physical smart objects for fault-tolerant, user-centric data transmission and communication.

10.13.4. Sensor Networks

CodeBlue [LMFJ+04] is an ad hoc sensor network infrastructure for emergency medical care. It is designed to provide routing, naming, discovery, and security for wireless medical sensors. The system is designed to enable continuous, real-time, noninvasive, wireless monitoring and tracking of patients. This is achieved by using an adaptive, multi-hop routing scheme for relaying vital data from patients to a wired base station (such as a PC or laptop) or directly to handheld devices carried by medical staff [MFJW+04]. In the process, CodeBlue requires a dedicated physical sensor network infrastructure, which is to be deployed in the area of activity, such as in a clinic and hospital environment, or in an ad hoc fashion at a mass casualty site, for instance. In contrast to such a sensor network approach, our system does not rely on dedicated sensor nodes as physical service infrastructure but instead makes use of existing, heterogeneous everyday smart objects that populate ubiquitous computing environments (e.g., mobile phones, personal digital assistants, notebook computers, and augmented objects in general that feature ad hoc and infrastructure-based communication capabilities). Further, we do not consider multi-hop communication schemes, but instead use smart objects with infrastructure-based communication capabilities as local communication gateways for mobile user devices.

IrisNet [GKK+03] is an Internet-scale sensor network service for resource-intensive data collection and processing. It relies on networked portable computers equipped with multimedia sensors (e.g., webcams) as so-called *sensing agents*, and a backbone of internetworked desktop computers as *organizing agents*. IrisNet aims to provide a software infrastructure for this platform, enabling users to query globally distributed collections of high-bit-rate sensors powerfully and efficiently. It envisions the distributed filtering, storing, and sharing of multimedia data on a planetary scale [CGN+05]. However, IrisNet does not consider support to resource-limited mobile devices for enabling fault-tolerant, user-centric data transmission or communication gateway services, as it is the case in our system.

10.13.5. Energy Efficiency of Mobile Devices

There has been considerable work on the energy efficiency of mobile, resource-limited devices with a focus on the collaborative relationship between the operating system and applications [FS99, WBB02, FS04] and system-level power management [RV03, SRS03]. We consider power-efficiency based on the prioritization of short-range ad hoc communication over long-distance infrastructure-based communication by using nearby smart objects with communication capabilities as local gateways.

10.13.6. Proxy-Based Recovery for Mobile Devices

Yao and Fuchs describe a proxy-based recovery scheme for wireless applications running on resource-limited mobile devices [YF00, YF01]. A *recovery proxy* is used to monitor the state of the connection of a mobile client to a remote back-end server. The proxy continuously maintains a copy of the client's state based on messages exchanged between the mobile client and the server. The proxy further

sustains the client's connection to the back-end server when a mobile client unexpectedly disconnects. The mobile client itself does not participate in checkpointing or message logging, thereby saving power, processor cycles, and bandwidth.

In our system, the virtual counterpart (VC) takes over similar tasks as part of the fault-tolerance management. In contrast to the approach by Yao and Fuchs where a proxy is non-persistent and mobile clients may migrate from one proxy to another, the applied VC concept in our system constitutes a permanent virtual representation with a unique address and identifier – the VC may outlive the physical representation of the mobile client. Further, in the recovery proxy approach, only state information related to the message exchange and data transfer between the mobile client and the back-end server is maintained and restored in the case of crash failures. The VC, however, provides additional fault-tolerance functionality by protecting the clients application state and configuration, and by serving as a intermediate for the remote reconfiguration of the fault-tolerance strategy on the mobile client.

Acknowledgments

The author wishes to thank Thomas Mazhuancherry [Maz03] and Christian Gegenschatz [Geg04] for their work on the implementation of the fault-tolerance mechanisms and of the mobile patient monitoring platform.

11. Super-Distributed RFID Tag Infrastructures

With the emerging mass production of very small, cheap Radio Frequency Identification (RFID) tags, it has become feasible to deploy such tags on a large scale. In this chapter, we advocate distribution schemes where passive RFID tags are deployed in vast quantities and in a highly redundant fashion over large areas or object surfaces. We show that such an approach opens up a whole spectrum of possibilities for creating novel RFID-based services and applications, including new means of cooperation between mobile physical entities. We also discuss a number of challenges related to this approach, such as the density and structure of tag distributions, and tag typing and clustering.

11.1. Super-Distribution of Radio Frequency Identification Tags

Passive RFID tags typically incorporate a miniature processing unit and a circuit for receiving power if the tag is brought within the field of an RFID reader. The tags are usually attached to mobile objects such as supermarket goods or other consumer products, and when interrogated they send their identity to the reader over distances ranging from a few centimeters up to a few meters, depending on the type of tag.

RFID tags that are spread across a particular space in large redundant quantities can in turn be regarded as a "super-distributed" collection of tiny, immobile smart objects. The term "super-distribution" refers to the fact that a vast number of tags are involved, similar to the notion of "super-distributed objects" in [OMG04]. Accordingly, we will refer to such a highly redundant tag distribution as a *super-distributed RFID tag infrastructure* (SDRI). However, in contrast to the concept of super distributed objects, which provides an abstraction for ad hoc communication between physical entities and devices as well as software components and services, we explicitly focus on the *physical distribution* of *physical entities* and on the delivery of location-aware services based on the resulting physical infrastructure.

A highly redundant and dense distribution of tiny objects is also a common characteristic of wireless sensor networks which consist of a large number of very compact, autonomous sensor nodes. However, the sensor network concept differs fundamentally from the SDRI approach: in contrast to a fixed structure of independent and passive tags as part of an SDRI, wireless sensor networks are based on the "collaborative effort of a large number of nodes" [ASSC02b]. Further, the topology of wireless sensor networks may change due to mobility on the part of its nodes. In addition, wireless sensor nodes carry their own power supply used to enable active sensing, data processing, and communication with other sensor

nodes, whereas passive RFID tags only have very limited functionality, generally restricted to the reading and writing of a small amount of data. Also, compared to typical wireless sensor networks with nodes communicating over distances of tens of meters or more, mobile RFID antennas generally operate at a much shorter range.

By deploying an SDRI in an area, the overall physical space is divided into tagged and thus uniquely identifiable physical locations. This means that each tag can be used as an identifier for the precise location it covers, where coverage is pragmatically defined as the reading range of the tag. What we thus obtain can be described as an approximate "discrete partitioning" of the physical space, of which different implementations are possible. If the tags of an SDRI are distributed according to a regular grid pattern, for example, then the partitioning of the physical space itself can be considered a physical grid of uniquely addressable cells approximating the concept of a regular occupancy grid, as applied in the field of mobile robot navigation [Elf89], for instance. If, on the other hand, the tags of the SDRI are distributed in a random fashion, we obtain an irregular pattern of uniquely addressable cells. Ideally, these cells are non-overlapping and cover the whole area. In practice, one can only approximate these properties.

In addition to the potentially massive redundancy of RFID tags, two particularly interesting qualities of SDRIs are that they enable mobile devices to interact with their local physical environment, and that such an interaction can be performed in a highly distributed and concurrent manner. In the following, we explain these qualities in more detail.

11.1.1. Local Interaction with Physical Places

An SDRI enables *local interaction with physical places* by allowing mobile objects to store and retrieve data in the precise geographic location in which they are situated (by writing to or reading from nearby RFID tags). Independent, anonymous entities are thus in a position to share knowledge and context information *in situ*.

One potential application of this quality is self-describing and self-announcing locations, where mobile devices can gain contextual or topological information on the spot simply by querying the local part of the RFID tag infrastructure. For instance, mobile GPS-enhanced vehicles could store position information in the very place where they obtained it while moving within an SDRI. Once a sufficiently large proportion of an affected area has been initialized in this manner, other mobile devices can be helped to recalibrate their GPS receivers, and GPS-less devices can use this location information in order to determine their position without using a dedicated positioning system themselves. Position information stored in the SDRI can also be used to establish a fall-back service in case the primary positioning service of a mobile device or vehicle is temporarily unavailable, thus increasing the overall availability of position information in the area. Further, an SDRI facilitates the definition of arbitrary regions within physical spaces: virtual zones, barriers and markers can easily be defined by marking particular tags (or the tags along a border line) in the SDRI accordingly.

Furthermore, by enabling mobile devices to leave data traces, messages, or links to virtual information (residing in a background infrastructure) wherever they roam, SDRIs in general support the creation of *physical anchor points* that can

serve as *local entry points into virtual spaces*. It is therefore possible to use the RFID tags of an SDRI as an alternative medium for implementing physical hyperlinks [Kin02, RB03] in virtual spaces, or as a means of attaching virtual annotations [SDHM03] to physical places.

By providing a means for roaming mobile objects to anchor and thus persistently store location-dependent sensor information on the spot, SDRIs also constitute a self-sufficient alternative to services such as GeoWiki [FJB+04] that link virtual information to geographical addresses. Here the use of an SDRI does *not* require explicit knowledge of the current geographic location nor the continuous availability of a dedicated location service with a high resolution and accuracy.

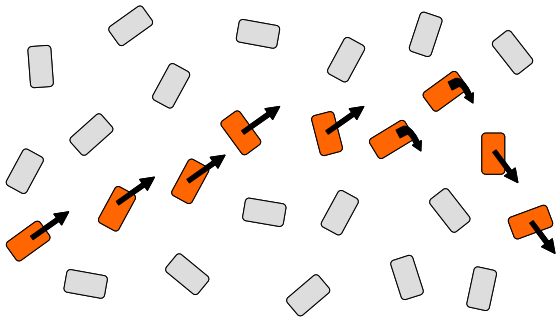

Figure 11.1.: A mobile object has left a virtual trace on an RFID tagged floor space. Each tag that is part of the trace contains information about the identity of the tracked physical entity and indicates changes in the direction of movement (orientation) of this entity

Mobile objects may also *leave virtual traces* in the physical space they traverse. By writing an anonymous ID to writable tags[1] in the floor space, a vehicle, for example, can leave a trace which can subsequently be retraced and followed by other mobile objects (see Fig. 11.1). These traces could later be overwritten by other vehicles or persons, so that they would fade away over time. By exploiting tag redundancy and enforcing suitable tag writing strategies, it should be possible to prevent the immediate deletion of a newly laid trace by following objects. For example, a moving vehicle can randomly choose one k tags out of n available tags per location in order to implement a redundant storing of trace identifiers. Furthermore, a single tag can store the IDs of several different traces (memory space permitting).

Further, as SDRIs provide mobile devices with the technical means of dispersing electronic information throughout the physical environment, they can be employed as a foundation for systems that have the goal of enabling *environment-mediated human-to-human communication* [Gel98].

11.1.2. Global Collaboration Between Mobile Objects

An SDRI can be regarded as a scalable shared medium with (almost) unlimited, independent, and highly distributed physical "access points". In this respect, an

[1] Tags are physically writable if an RFID scanner can be used to write data directly onto the tags. However, it is also possible to virtually write data onto read-only tags by means of a supporting background infrastructure, as described in Section 11.1.4.

SDRI is particularly conducive to supporting *global collaboration*, where several independent physical entities work together on a single task in a highly decentralized and concurrent fashion. This is possible since SDRIs enable mobile devices to store and access data locally at the precise location they occupy at a given moment, so that devices situated at different locations within an SDRI can read from or write to tags simultaneously and independently.

Consequently, SDRIs are well suited for the implementation of a number of collaborative applications. Mobile objects that gather location-dependent information can store that information directly at the respective location by means of the SDRI, resulting in teamwork between anonymous entities. Such a loosely coupled collaboration can be exploited for initializing or "bootstrapping" a particular SDRI with positional or topological information, for instance. Further, depending on their capabilities, mobile objects can participate *on the fly* to achieve such a common secondary goal while actually pursuing a different primary objective.

Another concrete example of collaboration is the collaborative exploration (mapping) of an uncharted territory. Since individual observations are based on globally unique tag IDs, the different map-making observations of independent mobile entities within a specific area can be unambiguously combined to form a global map. Such an SDRI-aided collaborative map-making process scales well as it can be performed by an arbitrary number of concurrent entities in a highly decentralized fashion.

Of course, for some of the scenarios described to become possible, both technical challenges (e.g., reading from or writing to RFID tags at high velocities) and conceptual issues (such as the question of commonly understandable topological models and ontologies, or the problem of updating information previously stored in the SDRI in order to reflect changes in the environment) have to be addressed.

11.1.3. Scope of Deployment

So far we have discussed a number of scenarios where SDRIs are deployed over large floor spaces in order to provide novel means of local interaction and global collaboration. However, the scope of SDRI deployment is not just limited to such large-scale scenarios. There are also various situations where smaller-scale SDRIs have their distinct benefits.

For a table equipped with RFID tags embedded in its surface, for example, the intrinsic qualities of SDRIs still apply: such a "small-scale" SDRI also yields uniquely addressable cells, allowing multiple devices (or users) to interact with different sections of the tabletop simultaneously. If the tag distribution of the tabletop is known or if each tag knows its position with respect to the local coordinate system of the table, a smart object on the table can also easily determine its position with regard to the tabletop, or derive the relative distance of proximate smart objects on top of the table by communicating and exchanging particular tag IDs or tag coordinates. Similarly, a wall whose surface is coated with a layer of RFID tags can be turned into a "smart notice-board", featuring support for the positioning of objects that are attached to it, and providing physical links to virtual information spaces to the users of the notice-board.

11.1.4. Physical vs. Virtual Tags

If RFID tags support read-write operation, then they obviously enable mobile devices to store a certain amount of data directly on the physical tags themselves. As a consequence, a mobile device can read from and write to the physical matter at its respective location, literally speaking.

Accordingly, if the available physical RFID tags are of a read-only type, a mobile device cannot directly write data to the tags. However, in this case we can still use the unique ID of the tags to unambiguously map each *physical tag* of the SDRI to a corresponding *virtual tag* residing in the background infrastructure. Rather than writing to a physical tag within the direct range of the mobile device, the device instead wirelessly connects to a virtual representation of the tag. The virtual tag may either simply provide the basic data read/write operations of a physical read-write tag, or even augment its capabilities by offering additional services which cannot be implemented on the small physical tags themselves due to resource limitations. One distinctive advantage of virtual tags over mere physical tags is that they don't suffer from physical resource limitations, enabling us to write an almost unlimited quantity of data to a virtual representation of the physical tag.

To ensure instantaneous read/write access to virtual tags, continuous wireless access to a background infrastructure that manages the virtually written tag-data is needed (if the virtual tag representations are not maintained on the mobile devices themselves that interact with the SDRI). In this case, the SDRI is no longer fully self-sufficient.

11.2. Efficient and Redundant Large-Scale Deployment of RFID Tags

The deployment of large quantities of tiny RFID tags over large areas necessitates an efficient means of tag distribution. Instead of minutely distributing tags across an area according to a rigid and well-defined pattern, which typically goes hand in hand with a time-consuming calibration of the tags, we think that a *highly redundant random distribution* of tags is often more favorable. Such a random distribution can be achieved in various ways. For instance, RFID tags could be randomly mixed with various building materials such as paint, floor screed, concrete, etc., which is similar to the idea of mixing computer particles with "bulk materials" as described by Abelson et al. [AAC+00]. Of course, in some cases such a procedure requires quite durable and resilient tags. But even if a certain percentage of tags were to be rendered defective in the process, the number of operable tags could be controlled by applying the necessary degree of redundancy.

If tags are uniformly distributed in a random manner over large areas, we can make assumptions about the average tag density and the coverage of the area. In some cases we can even randomly distribute tags and still maintain a certain regular distribution structure. By integrating RFID tags at regular intervals into carpet weaving thread, for instance, it is feasible to weave a complete carpet or produce carpet tiles that exhibited a regular RFID tag texture (e.g., forming a mesh of RFID tags as it is the case with the Vorwerk RFID carpet [Vor05]). Even though we would not know the absolute positions of the tags after an RFID-augmented carpet had

been laid out in a random fashion, we would still have precise information about the distances between neighboring tags and about the overall tag density.

Although a random large-scale tag distribution enables a dense and, on average, uniformly distributed coverage in a comparatively economic and straightforward fashion, the procedure does pose some challenges. For instance, if the costs per RFID tag are too high even in mass production, a dense large-scale deployment may not be economically feasible. Further issues are the durability of RFID tags that are embedded in a carrier material, and the "bootstrapping" of super-distributed RFID infrastructures with respect to positioning and the provisioning of location-dependent context information.

11.2.1. Deployment of RFID Readers vs. Tags

Rather than tagging large areas with small, cheap RFID tags, it is also possible to distribute RFID reader antennas instead. By integrating an array of stationary RFID antennas into the floor, as described by [ABO02], it is possible to detect tags that pass over particular readers, or even track certain tags if the output of several readers is combined and analyzed. Thus, by simply attaching a passive RFID tag to a mobile device, the latter is freed from the extra burden imposed by an energy-consuming RFID scanner.

However, such an approach has several disadvantages compared to the concept of SDRIs. First, integrating readers or antennas into the floor or into surfaces is resource-intensive, as quite a large quantity of expensive RFID readers and antennas is needed to achieve a good resolution and coverage. For the continuous operation of such RFID equipment, additional energized electronic devices are required. Furthermore, the deployment of the RFID equipment is complex and may require a considerable amount of construction work, not to mention the costly maintenance in the event that a device has to be replaced at a later point in time. Also, if the antennas are embedded into the environment and the mobile entities are tagged with passive RFID tags, the latter have only a limited means of controlling their degree of "visibility". The mobile objects cannot easily prevent themselves from being detected or tracked by the environment. If, on the other hand, the environment is being tagged and remains passive, the mobile entity itself is performing the sensing. Consequently, the mobile device remains in control of any interaction taking place with the environment, thus facilitating the implementation and enforcement of privacy policies, for example.

11.2.2. RFID Tag Distribution Patterns

In order to accomplish a large-scale distribution of RFID tags, there are a variety of possible tag distribution patterns to choose from. Typical distribution patterns include:

- *Random uniform distribution:* Tags are uniformly distributed over a certain area in a random manner.

- *Regular distribution:* Tags are distributed in a regular pattern, but usually with random tag identification numbers. Typical regular patterns are:

- *Grid pattern:* Given a grid with edge length d, each non-border tag has four nearest adjacent neighbors at a distance d and four farther adjacent diagonal neighbors at a distance $sqrt(2) * d$.
- *Equilateral triangulation pattern:* Each tag has six equidistant neighbors at a distance d.

From the perspective of a mobile object, random tag distribution patterns have inherently different properties compared with regular patterns. One example is illustrated in Fig. 11.2. We expect there to be other generally advantageous patterns for the distribution of RFID tags, such as irregular but non-random distribution patterns, for example. This calls for the investigation of beneficial tag distribution patterns and their respective properties as part of future research.

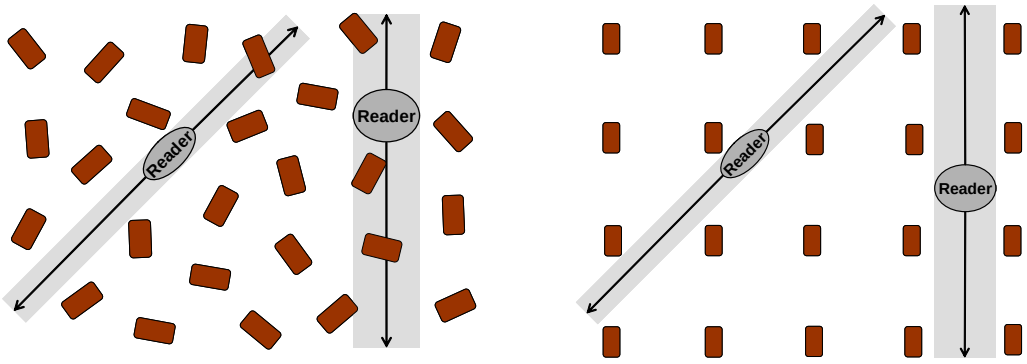

Figure 11.2.: In an area of randomly and uniformly distributed tags, a mobile reader will come across another tag while moving on a straight path with probability $p > 0$ depending on the tag density and the distance traveled. In regular RFID tag distributions, such as a grid structure, it is possible that a mobile reader may choose a straight path where it will never encounter a single tag

11.2.3. Sparse vs. Dense Tag Distributions

The maximum density of an RFID tag distribution that can be achieved in an SDRI primarily depends on the properties of the underlying RFID technology. A secondary aspect is the required degree of resolution and the degree of tag redundancy, which can be expressed by the average number of tags that are within the range of the mobile reader antenna at an arbitrary location, for example.

Sparse Non-Overlay Tag Distribution

If the RFID technology used for the realization of an SDRI does not support collision detection and resolution, only a single tag should be within the range of the reader antenna at any given location. We call this a *sparse* tag distribution. In this case, the maximum tolerable tag density of the SDRI is limited by the characteristics of the available reader antenna. The distributed tags should be spaced in such a way that each tag exclusively covers an area (typically larger than the scan range covered by the reader antenna). Otherwise, the tags cannot be reliably

scanned due to frequently occurring collisions. But even if collision resolution is available, one might deliberately prefer a non-redundant RFID tag distribution in order to simplify or speed up tag processing.

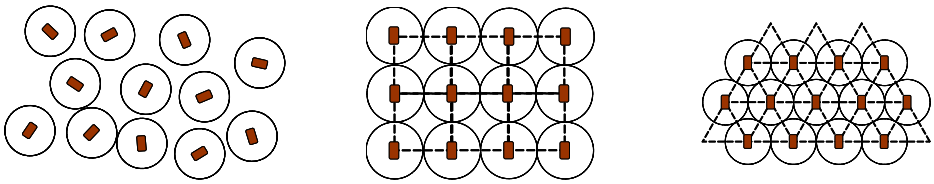

Figure 11.3.: Examples of sparse non-overlay tag distributions: a) random b) grid c) triangular. The circles enclosing the tags indicate the idealized area in which a specific tag can be detected

However, a sparse, non-overlapping tag distribution has several disadvantages. Firstly, it results in an inflexible partitioning of the covered area into coarse-grained cells whose dimensions are defined by the range of the mobile reader antenna. Secondly, the antenna fields of RFID antennas projected onto a planar surface are often circular, elliptical, or even irregular. As a consequence, in these cases a non-overlapping tag distribution would not cover the entire area, but would result in "blind spots" at the fringes where no tags will be detected at all by such antennas (see Fig. 11.3). Thirdly, a non-overlapping tag distribution is not redundant, which means that tag failures cannot be compensated. Even if blind spots at the fringes of non-overlapping detection ranges can be regarded as negligible transition zones and therefore as acceptable, the failure of single tags would result in ever larger areas where no tag can be detected at all.

Dense Overlay Tag Distribution

In order to establish an SDRI with overlapping tag scan ranges, so that multiple tags can be detected per scan and per location, the RFID system must support collision arbitration or resolution. In this case, the tolerable tag density is limited firstly by the technically viable proximity of tags at which these tags still respond correctly during a scan (and are unaffected by *tag detuning* [FL04], for instance), and secondly by the maximum number of tags within antenna range that can be detected simultaneously and in a reasonably short time by the anti-collision scheme. Both factors depend on the particular characteristics of the RFID system.

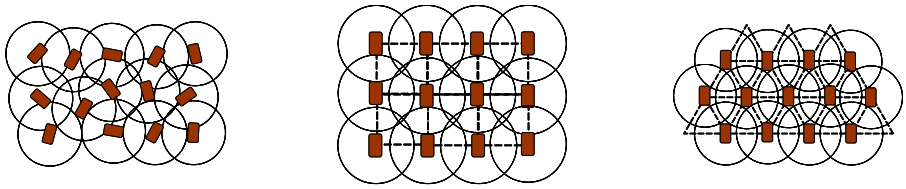

Figure 11.4.: Examples of dense overlay tag distributions: a) random b) grid c) triangular

Dense overlay tag distributions allow us to achieve a fine-grained and complete tag coverage without "blind spots" (see Fig. 11.4). An important advantage is

the introduction of redundancy with respect to the number of RFID tags that are detectable per scan at a given physical location. By using an anti-collision system and a sufficiently high scan range, we can detect several tags per location. A location can then be described by the set of all the tags that are detectable at the respective place. If we detect several tags, we can calculate the center point of their locations and use that value as a position approximation for the current location by using localization techniques as proposed in [BHE00], for instance. This has the benefit of increased robustness with regard to tag failures: if a tag is destroyed or fails over time, we still detect the remaining functional tags at a location.

11.2.4. Tag Typing and Clustering

Even though each RFID tag in an SDRI has its own unique ID, in certain situations it may be sufficient to discern only particular categories or types of tags. For the large-scale deployment of RFID tags within a building, for example, one might want to use different types of tags in different sections of a building, such as a tag type A for corridors, a type B for public spaces, a type C for private areas, and another type D to mark stairways and elevators. So in addition to a unique tag ID, each tag could also be equipped with an additional data field containing a predefined type identifier. If the RFID tags supported physical (or virtual) write access, it would be possible to "impregnate" the desired type identifiers onto tags after their distribution. Alternatively, tags of a particular type can be grouped and pre-packed together for efficient deployment.

There are many different fields of application for tag typing and clustering, such as marking potentially hazardous areas for visually impaired people who are equipped with a smart cane[2] (e.g., to warn the person about an approaching stairway), or defining different area categories for mobile robots (e.g., virtual barriers).

11.3. Initial Prototype Development and Assessment

As part of a first practical assessment of the feasibility and versatility of our approach, we implemented the prototypes of two SDRI-based applications using the Hitachi mu-chip [Hit06] and LEGO Mindstorms [LEG06]. The first prototype is a location-aware autonomous vacuum cleaner [Dom04] equipped with a mobile RFID reader and antenna, which adjusts its behavior based on tags embedded in the floor space at its particular location, such as avoiding areas that are marked as off-limits or staying within an area surrounded by a virtual barrier (consisting of tags that have been marked accordingly in the teaching mode of the mobile robot).

The second explorative prototype system was a system for collaborative map making where independent mobile vehicles (each again equipped with a mobile RFID reader and antenna) explore a previously unknown area (see Fig. 11.5). Starting from a known position, each vehicle chooses a random path through the

[2] Goto et al. developed a smart cane with an integrated RFID reader, for instance. It responds to RFID tags which serve as "data-carriers" embedded in the floor at places of interest [GHM99]. Here an SDRI could provide a fine-grained and complete distribution of data-carriers over large areas, as opposed to the cumbersome process of deploying such data-carriers selectively.

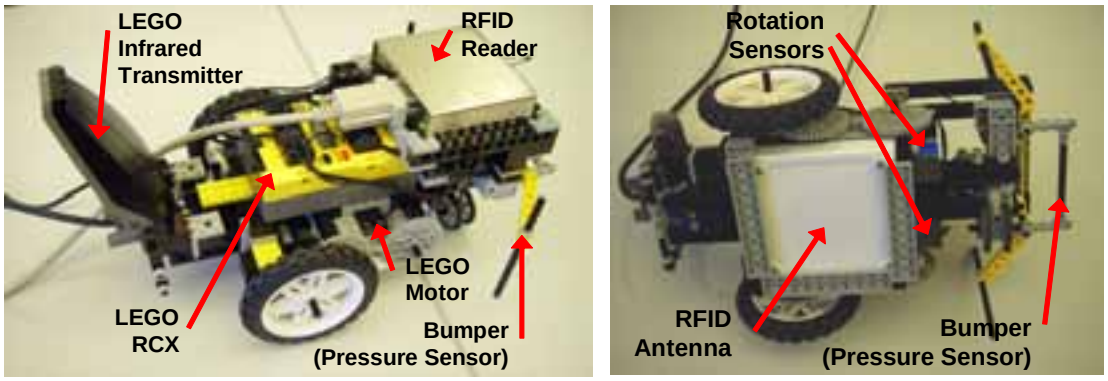

Figure 11.5.: Mobile model vehicle built using the LEGO Mindstorms [LEG06] technology. The vehicle motor and sensors are operated by the LEGO RCX unit, which receives control commands from or sends status information back to the LEGO infrared transmitter which is connected to a laptop computer. The RFID reader is also connected to the laptop on which the software for reading tags and calculating the tag positions is executed. The bottom view of the vehicle shows the RFID antenna, which is mounted approx. 1 cm above the floor, and the two rotation sensors measuring the revolutions of the back wheels. At the front of the vehicle, a bumper is connected to a pressure sensor to detect collisions while the vehicle is in motion

area and keeps track of the tags encountered and the relative inter-tag distances. The separate tag observations are subsequently merged by means of an efficient least-squares coordinate transformation algorithm, yielding a global map containing the absolute positions of known RFID tags. This map can then be used by other mobile devices for navigation and self-positioning purposes. For the map-making trials, a sparse random tag distribution on a 0.5×0.5 m floor area has been used. The mobile reader antenna covers an area of about 9×6 cm at a distance of about 1 cm above the floor. Using mu-chips equipped with a 4 cm long film antenna (inlet), a density of about 120 tags/m^2 has proven to be sufficient for this type of application.

We further investigated and demonstrated the effectiveness of the super-distribution concept with regard to small-scale application scenarios. For instance, we prototypically implemented and evaluated a system for the secure reconstruction of paper documents that are densely tagged with RFID tags [Pyt04], and an RFID-augmented jigsaw puzzle game [Boh04b].

11.4. Conclusion

Based on our initial experiences described in Sect. 11.3, we decided to perform further experiments to explore different tag distribution patterns and tag densities in practice and to determine their influence on scalability, efficiency, and robustness. Apart from building specific SDRI-based applications for demonstration and more systematic evaluation purposes, we also used this work as a starting point for developing a general middleware that provides an efficient and reliable means

of accessing an underlying physical SDRI. We investigated fault-tolerance aspects such as fault-tolerant read/write operations, an automated maintenance procedure for the "hot" integration of newly distributed tags during operation, and high-level services such as location management, self-positioning, and local data sharing. We were particularly interested in the issue of robustness and the degree of fault tolerance that can be achieved through massive redundancy of "super-distributed" RFID tags. The resulting fault-tolerant service middleware based on super-distributed smart entities and its prototypical reference implementation are presented in the following chapters.

Acknowledgments

The location-aware autonomous vacuum cleaner was built by Svetlana Domnitcheva, Julio Perez, and Matthias Sala [Dom04]. We would further like to thank Marco Bär for his work on the collaborative map-making prototype [Bär04], and Hitachi SDL, Japan, for providing us with mu-chips and tag readers.

12. A Fault-Tolerant Service Middleware for Location-Aware Systems Based on Super-Distributed Smart Entities

In the previous chapter, we presented distribution schemes where passive RFID tags were super-distributed (i.e., deployed in vast quantities and in a highly redundant fashion) over large areas and object surfaces. In this chapter, we introduce the concept of *super-distribution of smart entities* as a generalization of the concept of super-distributed RFID tag infrastructures, with the goal of providing physical infrastructure support for the development of reliable and highly available location-aware applications. We outline a middleware that provides a set of fundamental location-dependent services where fault tolerance is achieved on two levels. Firstly, the abundance of smart entities is exploited to compensate for the failure of individual smart entities through local data replication, data fusion, and abstractions. Secondly, the localized interaction of mobile devices with fixed smart entities in the environment allows for the realization of dependable services that remain operational even in the case of physical damage in other areas of the infrastructure, and in the absence of network connectivity and of remote services.

12.1. Dependable Location-Aware Services for Mobile Devices

In recent years, location information has become increasingly important for various mobile and portable devices, opening diverse application fields [MB03]. Put in general terms, location information can be used to provide services whose execution can be dynamically adapted to the characteristics of the user's particular context at his or her current location [CLMZ03]. At the same time, as a result of ongoing advances in miniaturization and electronic components, it has become feasible to densely distribute small computerized entities in large quantities in physical environments or over object surfaces, as we saw earlier in the example of super-distributed RFID tags in Chapter 11.

Based on these observations and results, we motivate *super-distribution* as a *general design principle* for the realization of reliable and highly available location-dependent services for mobile devices. The main idea is to distribute small computerized physical objects in large quantities over object surfaces, such as across floor spaces or walls, to obtain a dense and highly redundant distribution of "smart

entities". The resulting smart entity infrastructure then forms the basis for the development of a *fault-tolerant middleware* that provides a set of fundamental location-aware services and abstractions. In return, this middleware serves as a foundation for the development of dependable location-aware mobile applications.

12.2. Middleware Support for Super-Distributed Infrastructures

12.2.1. Super-Distribution of Smart Entities

We define a *smart entity* (SE) as a physical artifact that is enhanced by embedded computing technology of some kind. As minimum requirements of a SE we demand that it has a globally unique identifier and a read/write memory, both of which can be accessed by mobile user devices via wireless ad hoc communication. Examples for SEs are simple passive radio frequency identification (RFID) tags as well as self-powered embedded sensor nodes.

In accordance to our earlier definition of super-distribution of RFID tags in Sect. 11.1, we define *super-distribution of smart entities* as the process of deploying and distributing SEs in a dense, highly redundant (or abundant) fashion over some physical space. We call the resulting infrastructure a *super-distributed smart-entity infrastructure*. In this context, *dense* means that a mobile device that moves within a SE infrastructure will always find at least one SE within ad hoc communication range for in situ interaction or cooperation. *Highly redundant* means that the degree of redundancy is not deliberately set to a fixed value, but determined by an abundance of entities found in any physical location of the infrastructure. *Abundance* refers to a quantity of entities that significantly outnumbers the amount of entities that, ideally, would be required for a non-fault-tolerant, non-redundant operation of SE-infrastructure-based services in the absence of disturbances. In the following, we assume that the individual SEs are permanently attached to a substrate (e.g., floor spaces or object surfaces) at a physical place that is well-defined within a local or global coordinate system.

For applications that support mobile users, interaction with the super-distributed smart-entity infrastructure is performed by means of a *mobile device* (MoD). The MoD features a wireless communication interface, which enables it to communicate in an ad hoc fashion with SEs in its immediate vicinity. On each MoD, an independent instance of the SE infrastructure service middleware is installed and executed. A MoD can be carried by a user, or it may be part of other devices, such as being integrated into a mobile phone, a vehicle, or a blind man's stick, for example. In our model, interaction exclusively takes place between MoDs and SEs. At this time we do not consider interaction to take place between the distributed SEs themselves.

12.2.2. Middleware Support

The dense distribution of smart computerized entities across object surfaces (to serve as the foundation of a super-distributed service infrastructure) is new in the sense that its technical and economical feasibility has only recently evolved

owing to technological advances in microelectronics and embedded computing (cf. Appendix B.3). As a consequence, so far there is little system support for the development of applications and services based on such a physical infrastructure.

While in general the ad hoc creation of specialized applications based on super-distributed smart entities is always possible, such a procedure is bound to lead to closed systems with a narrow, limited scope and functionality. To support the rapid development of whole classes of applications, a fundamental challenge is to provide a set of general, basic services that satisfy a broad spectrum of requirements and needs. The development of reusable, basic services has the advantage that it greatly facilitates the development of higher-level services and applications.

In order to identify location-dependent services that particularly benefit from the availability of a super-distributed smart-entity infrastructure, we performed an analysis of the needs of various ubiquitous computing projects [Pir04]. We further identified a number of basic building blocks that encapsulate the hardware-specific aspects of the underlying physical infrastructure, provide low-level services and abstractions, and mask the complexities of fundamental maintenance and fault-tolerance management tasks from higher-level services. In a second step, we combined the various services into a layered middleware framework, according to their respective levels of specialization and universality. The resulting middleware architecture will be discussed in detail in Section 12.4.

12.2.3. Middleware Employment

From the user's perspective, interaction with the SE infrastructure is performed by means of a *mobile device* (MoD), which features a wireless communication interface for communicating in an ad hoc fashion with SEs in its immediate vicinity. On each MoD, an independent instance of the service middleware is installed, executing the individual middleware services as separate processes. Services can be turned on or off and configured separately. A MoD can be carried by a user or may be part of other devices, such as being integrated into a vehicle or into a blind man's stick, for example.

The execution of the service middleware on the MoDs themselves (rather than providing the services as part of a fixed background infrastructure) empowers the devices to interact with the super-distributed smart-entity infrastructure in an autonomous fashion. In particular, by maintaining information in SEs at the physical places where it is required, middleware services on a MoD can remain operational even in the case of physical damage in other areas of the infrastructure, and in the absence of network connectivity or the unavailability of remote services.

12.2.4. Dependability Issues

For the design and development of our middleware, dependability and fault-tolerance aspects played an important role at two different levels of abstraction: Firstly, the middleware services themselves had to be robust and fault-tolerant. On the one hand, they had to be able to tolerate failures of individual SEs during operation. On the other hand, these services had to be capable of integrating newly distributed SEs with minimal user intervention, to achieve a high serviceability (i.e., the ability to undergo maintenance without shutting down the system), and to minimize

maintenance overhead. Secondly, the middleware should enable the development and the operation of reliable and highly available location-aware applications. Here the main goal was to exploit the *locality* aspect of a super-distributed smart-entity infrastructure in order to provide basic services in situ which still function in the absence of network connectivity and background computing infrastructures.

12.2.5. Design Goals

An important conceptual design aspect of the realization of a middleware based on an underlying super-distributed smart-entity infrastructure for providing robust and reliable location-dependent services to mobile devices was the masking of the complexity stemming from fault-tolerance management, SE hardware access, and internal maintenance tasks. For that, it was important to decouple the operation of the middleware from the availability of individual SEs. The aim here was to compensate for single faulty SEs without losing the vital information linked to them, and without disrupting the services that make use of these entities. More specifically, we identified and implemented the following middleware design goals:

- *Hardware Abstraction:* The technology-dependent aspects of the underlying SEs are hidden from higher level services and applications, and abstractions that are independent from individual physical SEs are provided.
- *Fault Tolerance:* The redundancy resulting from the super-distribution of SEs is exploited for the realization of fault-tolerant basic services and operations.
- *Self-Organization and Self-Calibration:* The services offered by the middleware can perform initialization and calibration tasks autonomously without explicit user involvement.
- *Self-Healing Capabilities:* Maintenance tasks are performed autonomously, reducing the need for manual intervention and servicing to a minimum. For instance, the middleware supports the integration of additional SEs that are distributed at a later point in time and without a noticeable service interruption.
- *Transparency:* The complexities of fault tolerance, self-organization, and self-healing mechanisms as well as hardware-specific details are masked from higher-level services and applications.
- *Disconnected Operation:* The middleware services maintain a certain functionality even in the absence of network connectivity or during the unavailability of remote networked services.
- *Extensibility:* The middleware architecture is modular and extensible, facilitating the integration of additional services.

12.3. Motivating Usage Scenarios

In the following, we present scenarios that highlight some opportunities and advantages of services based on a super-distributed smart-entity infrastructure from the perspective of the user.

12.3.1. User-Centric Location-Dependent Services

Within a super-distributed smart-entity infrastructure, users directly interact with the SEs at their current location. For geographic guidance and navigation, a MoD can determine its current position by calculating an estimate from the individual positions stored on nearby SEs, or simply look up the current location with the help of a local map containing the positions of individual SEs. Users are able to share information about local points of interest or leave personal messages directly in the physical places where the information is most helpful and required. Super-distributed public directories (i.e., directories whose entries are physically distributed across the SEs of the SE infrastructure) provide localized information about room numbers or names of departments, offices, or personnel, enabling visitors to find their way unassistedly even in unfamiliar places and buildings. Besides, users can leave virtual data traces in an ad hoc fashion on the SEs passed along the way, permitting friends or colleagues to follow at a later point in time to places where an activity or meeting is to take place. By integrating the MoD with a blind man's stick or with a wheelchair, these services can also be made available to visually impaired people or to persons with walking disabilities. In addition, this enables these users to share information tailored to their particular needs in situ, such as information about nearby obstacles, dangerous crossings, and wheelchair-accessible ramps and gangways.

12.3.2. Dependable and Safety-Critical Services

Disasters caused by natural or human factors (e.g., earthquakes, large fire incidents, or terrorist attacks) often lead to the failure of infrastructure services in buildings or public places, disrupting electricity and communication networks. However, by maintaining safety-critical information in SEs at the physical places where it is required by rescuers, services based on super-distributed smart-entity infrastructures remain operational in physically intact areas of the super-distributed smart-entity infrastructure even when conventional background service infrastructures collapse or other areas are damaged, as the MoD directly interacts with local SEs via short-range ad hoc communication. In addition, a super-distributed smart-entity infrastructure allows for the provisioning of dedicated emergency services. For instance, by means of permanent virtual data traces stored on the SEs, it is possible to provide services that direct users (including professionals such as firefighters and emergency physicians) to the nearest emergency exit or life saving equipment.

12.3.3. Systems Support for Collaborative Activities

In some cases the activities of individual MoDs can be combined to contribute to a global task. For instance, MoDs that have a third-party positioning service (such as GPS) at their disposal can store obtained position readings on the SEs at their respective places. Thus the super-distributed smart-entity infrastructure can be "bootstrapped" with position information over time through a collaborative effort. Another example for collaboration is the construction of a global site map of a super-distributed smart-entity infrastructure by joining partial SE mappings obtained from individual MoDs. A super-distributed smart-entity infrastructure can also serve as a vast communal information space: the individual contributions of

users in different physical places can contribute to the creation of open, community-driven information services and directories.

12.4. Middleware Architecture

Based on our analysis of fundamental location-aware services that may be supported by a super-distributed smart-entity infrastructure (see also Sect. 12.2.2), we developed the layered service middleware architecture depicted in Fig. 12.1. While this list of middleware services is not exhaustive, we consider it to be general enough to accommodate a broad range of location-aware systems and applications. By means of our chosen modular approach, additional services can easily be added at a later point in time. The responsibilities of the different layers and the basic capabilities of their respective services are described in the following sections.

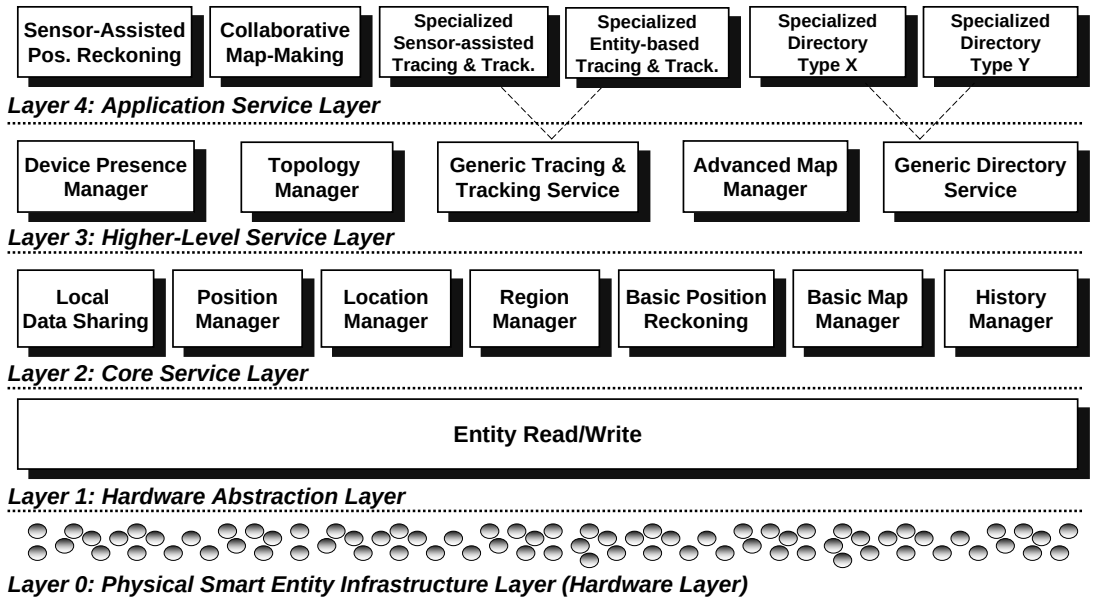

Figure 12.1.: Overview of the middleware for super-distributed smart entities

12.4.1. Hardware Abstraction Layer

The distributed physical SEs in their entirety constitute the physical SE infrastructure, which we refer to as *Hardware Layer* or Layer 0.

The *Hardware Abstraction Layer* (Layer 1) defines a generic and unifying interface to the underlying physical SE infrastructure, which is represented by the *Entity Read/Write* service. Middleware components on higher levels interact with the physical SEs exclusively through this service, which offers methods for discovering, writing data to, and reading data from nearby SEs. It also enables higher-level middleware components to write either physically onto the present SEs, or virtually, using a remote persistent memory space residing in the background infrastructure. The latter makes it possible to provide resource-limited SEs with arbitrary, potentially unlimited online content and memory space.

12.4.2. Core Service Layer

The *Core Service Layer* (Layer 2) consists of basic abstractions and generic services that operate on the physical SE infrastructure through the Hardware Abstraction Layer:

Local Data Sharing: The Local Data Sharing service enables a MoD to share data in physical places of the super-distributed smart-entity infrastructure with other MoDs. Concretely, it offers methods for the fault-tolerant storing and retrieval of data by writing to/reading from nearby SEs (Fig. 12.2(1)). Fault tolerance is achieved by replicating information across multiple local entities, with a configurable replication degree. As a background maintenance job, the Local Data Sharing service automatically detects newly added SEs and, if necessary, replicates data from nearby SEs according to the effective replication settings.

Location Manager: The Location Manager provides a MoD with methods to define and resolve abstract locations: a *Location* is defined by the set of stationary SEs that are detected in a specified range in the corresponding physical place, and by a unique identifier (Fig. 12.2(2)). Thus defined *Locations* are decoupled from the existence of individual physical entities that typically have a limited (or shorter) life-time. *Locations* can overlap to a certain degree, which means that individual SEs may belong to several *Locations* at the same time. The valid *Location* at a particular physical place is then determined by finding the previously defined *Location* whose set of SEs has the greatest overlap with the set of locally detected SEs (Fig. 12.2(3)). Taken together, *Locations* form a logical partitioning of a SE infrastructure. *Location* identifiers are directly stored on the defining SEs themselves. This enables other MoDs to identify *Locations* that have previously been defined at a physical place.

Region Manager: The Region Manager provides an abstraction for regions, which enables the logical description of geographic areas of arbitrary shape, based on the *Location* abstraction. A *Region* is defined as a collection of *Locations* and is assigned an arbitrary symbolic name. The Region Manager also provides methods for creating and detecting *Regions*, and an event-based notification service where clients can subscribe to occurring *Region* entering/leaving events. A global instance of the Region Manager can be used to make defined *Regions* known to other MoDs in the super-distributed smart-entity infrastructure.

Position Manager: The Position Manager offers methods that write position information (global or local position coordinates) to or retrieve it from physically nearby SEs. A position is calculated in a well-defined way (e.g., as the arithmetic mean) from the single positions of all SEs in range of the MoD executing the service (Fig. 12.2(5)). New position information to be stored on a SE is merged with the current position information (e.g., by calculating a new weighted arithmetic mean), respecting the number of previously combined position values (Fig. 12.2(4)). As a background maintenance job, newly detected SEs are initialized with available position information.

Basic Position Reckoning: Based on knowledge about the stationary SE distribution (such as the average distance between entities), Basic Position Reckoning service is responsible for providing basic information on the motion of the MoD running the service, such as estimations for the speed and traveled distance of the MoD.

Basic Map Manager: The Basic Map Manager service creates geographic maps of individual SEs based on position information obtained from the Position Manager or Position Reckoning services. Partial map observations can be sent to a global service instance residing in the background network infrastructure. A global Basic Map Manager instance combines partial maps into a single global map of the super-distributed smart-entity infrastructure. It also performs maintenance operations, such as the integration of newly discovered entities in previously charted areas, as well as the removal of defunct entities which were no longer detected during subsequent mapping passes.

History Manager: The History Manager allows the user to keep track of certain activities performed while using the middleware. For that, each middleware service component can insert logging statements (history events) into the executable code where applicable. Later, during operation, logging can be activated or deactivated for individual services by the user. The History Manager offers methods for retrieving logged history events for a specified service and a given time interval. For instance, a user may be interested in the logged readings of the Position Manager service in order to be able to backtrack his path later on when searching for a personal item that got lost or was left behind somewhere along the way.

12.4.3. Higher-Level Service Layer

The *Higher-Level Service Layer* (Layer 3) is represented by a collection of more specialized services and service templates that are built on top of the core services. Typically these higher-level services do not directly operate on the underlying SE infrastructure, but use the abstractions provided by the core services instead. We identified the following higher-level services:

Device Presence Manager: MoDs can register with the Device Presence Manager service and thus indirectly inform other MoDs about their names, current positions, or locations. The Device Presence Manager offers methods for registering (lease-based registration) and looking up MoDs, and for calculating distances between MoDs (based on their *Locations* or positions).

Advanced Map Manager: The Advanced Map Manager manages a geographic representation of the environment based on abstract *Locations* and *Regions* that were encountered during the movement of the MoD performing the mapping. Individual mappings can be shared among multiple MoDs. A global service instance manages individual map representations of a super-distributed smart-entity infrastructure, combines them to obtain a global map model of the area, and performs map maintenance operations such as the removal of disbanded region definitions.

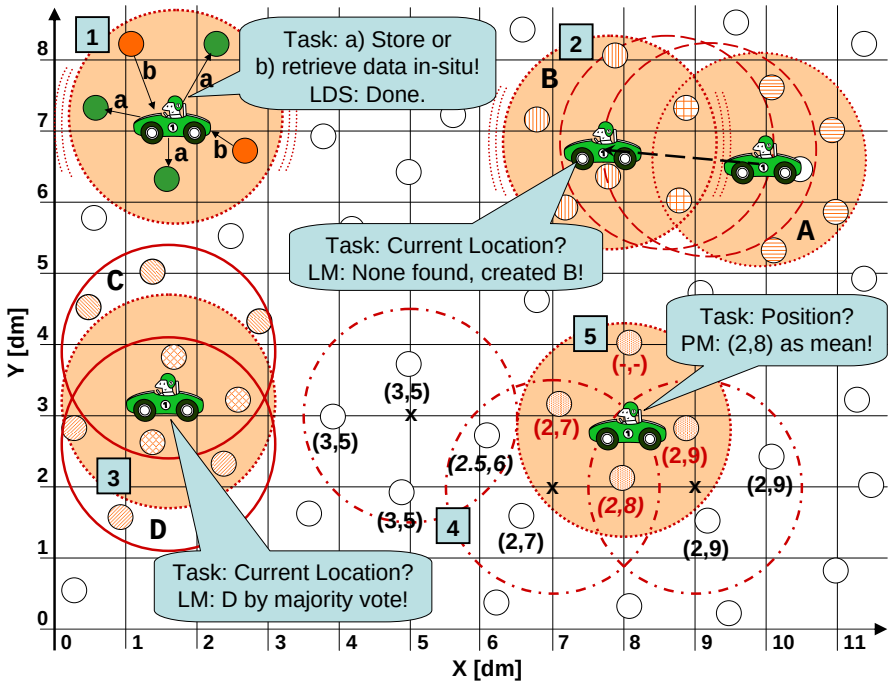

Figure 12.2.: Functionality of exemplary middleware services: (1) LDS: local data sharing; (2) Location Manager (LM): defining a new *Location*; (3) LM: resolving the current *Location*; (4) Position Manager (PM): SEs initialized with position information; (5) PM: calculating an estimate for the current position

Topology Manager: The Topology Manager service features an API and a graphical front end for defining, maintaining, and visualizing *Regions* based on a map model of the physical environment.

Generic Tracing and Tracking: The Generic Tracing and Tracking service provides a generic framework for placing and detecting data traces. A data trace consists of a sequence of data objects containing a trace identifier and a timestamp. The data objects are stored on the physical and/or virtual memory of SEs found along the way (tracing mode). A user can reveal a personal trace identifier to other users, enabling them to use the Generic Tracing and Tracking service for following the trace (tracking mode).

Generic Directory: Directory services have been recognized as a suitable means for providing location information to location-aware applications [Maa98]. In our middleware architecture, we introduced a Generic Directory service which combines localized information access within the super-distributed smart-entity infrastructure with directory functionality. For that, the service defines a generic interface for searching, reading, comparing, adding, modifying, and deleting entries stored on local SEs.

12.4.4. Application Service Layer

In the *Application Service Layer* (Layer 4), we find application-specific services and specialized instantiations of service templates.

Examples for services on the application level are specialized directory instantiations, such as directories for room facility management or personnel within an office environment, or directories with information on shops, entertainment facilities and restaurants within a shopping mall. Further examples are enhancements to middleware services that require additional hardware or third-party services, such as sensor-assisted tracing/tracking and position reckoning services. Examples for location-dependent applications based on the SE infrastructure middleware are navigation systems, which may be based on the Position Manager or Map Manager services, for instance.

On the application level, it is also feasible to combine different middleware services for the delivery of services tailored to specific user groups. For instance, the positioning and map-making capabilities of the super-distributed smart-entity infrastructure can be advantageously combined with directory services to provide customized services for persons with disabilities. In general, the middleware services can be favorably integrated within systems that aim at improving the perception of physical surroundings and the interaction with physical locations, such as with the Chatty Environment [CKR04], a system which provides information on physical places to visually impaired users by means of speech output and other feedback mechanisms.

12.5. Middleware Design Aspects

12.5.1. Extensibility

Our middleware, which has been implemented as a prototype (see Sect. 12.6), is based on an open and modular model: additional services are simply added to one of the different layers as new separate components, according to the particular dependencies and the degree of specialization.

12.5.2. Smart Entity Data Management

Some middleware services need to store data on SEs. This has to be performed in a structured way in order to make it possible to discover and retrieve specific data at a later point in time. To enable a basic form of data management, our prototype is based on the concepts of Service Data Units (SDUs) and Smart Entity Directories, which emulate a basic file system. An SDU encapsulates the data which is required by a certain service to be stored on a single SE for a well-defined purpose. In other words, an SDU represents a service-specific atomic data unit. Each SDU further contains a descriptor which provides information on the type and amount of data it contains, and meta information (persistence flag, dirty flag, CRC checksum). A Smart Entity Directory lists all SDUs that are available on a particular SE and shows the remaining amount of memory space.

12.5.3. Physical vs. Virtual Memory

Ideally, the Hardware Abstraction Layer and thus all higher middleware services using this layer operate on the physical memory of the SE infrastructure, which we call *physical* SE memory access. In practice, SEs only have limited resources at their

disposal, which in particular applies to storage capabilities. Further, depending on the underlying technology for memory storage and wireless communication, the physical read and write access may be error prone and slow. For these reasons, we have designed the Entity Read/Write service to also support *virtual* SE memory access operations.

The virtual SE memory is maintained in a database residing in the background network infrastructure or in the Internet, either as a domain-specific or as a global service. The virtual memory not only mirrors the physical memory of the corresponding SE, but also provides additional, virtually unlimited storage space. Access to the virtual memory of a SE is only possible when that entity is physically present within communication range. Whenever the physical memory of a SE is modified (i.e., when a service writes one or more data items to the SE), its virtual representation is updated accordingly.

12.5.4. Fault-Tolerance Management

In our approach, SEs represent the basic abstractions and elementary units of the super-distributed infrastructure. The temporary unavailability or permanent failure of SEs therefore is the primary source of faults that has to be dealt with by the middleware.

By design, redundancy is an inherent quality of SE infrastructures, based on the distribution of abundant quantities of SEs. In our middleware, this abundance is instrumented in three ways for achieving fault tolerance:

Local Replication: The *locality* aspect of a SE infrastructure makes it possible to locally replicate service data units required for the operation of middleware services over multiple SEs. Thus it is possible to locally increase the number of replicas of a specific SDU from 1 to N, where N may be defined statically or calculated dynamically according to the local SE density and a given replication ratio. The maximum value of N depends on the density of the SE distribution and on the communication range of the MoD interacting with the SEs. If N-fold replication is technically feasible, the failure of up to $N - 1$ individual SEs can be tolerated by middleware services using local data replication.

Local Data Fusion: Fault tolerance may also be achieved by means of *data fusion* instead of replication, if appropriate. For instance, it is possible for the Position Manager to interpolate the position coordinates at a certain physical location by calculating the mean of the individual positions retrieved from nearby SEs. In this case, the absence or failure of single SEs does not lead to the failure of the service, but only to a gradual and acceptable degradation of its quality, as long as the density of the SE distribution does not drop below a critical value.

Abstractions: Fault tolerance is further achieved by providing persistent fundamental *abstractions* which decouple the interaction between services and the SE infrastructure from individual physical SEs, which are potentially short-lived and prone to failures. We have introduced two such abstractions in

the Core Service Layer, called *Location* and *Region*. These abstractions remove the dependence of services from single physical entities by mapping geographic places to redundant sets of SEs rather than to individual SEs. Thus it is possible to persistently link information and services to well-defined abstract locations and regions rather than to individual entities which are liable to fail and become unavailable over time.

12.5.5. Smart Entity Infrastructure Maintenance

The individual SEs of a SE infrastructure are liable to fail over time. In the long run, this leads to a deterioration of the infrastructure as the density of the SE distribution decreases. A direct consequence hereof is that the replication degree of data decreases accordingly. Therefore, in order to preserve the fault-tolerance degree of the middleware services based on the availability of redundant SEs as part of the super-distributed smart-entity infrastructure, some form of maintenance is required.

In particular, to keep up the SE density of the SE infrastructure and thus to sustain readiness of operation, additional SEs have to be replenished (i.e., redistributed) from time to time. Here a major challenge is to support the integration of newly distributed SEs during runtime, which we refer to as *hot-integration* of SEs. The way in which such a hot-integration can be performed depends on whether a middleware service is *stateful*, that is it does require data to be stored on the SEs themselves during operation, or *stateless*, if it does not depend on such data.

Maintenance of Stateful Services. In our middleware system, stateful services perform maintenance and repair tasks in an independent and autonomous manner, thus displaying *self-maintenance* and *self-healing* capabilities:

Replication Maintenance: Services that use replication continuously monitor and, if necessary, restore the desired degree of data replication per SDU (while the MoD is idle). The Local Data Sharing service, for instance, performs regular replication maintenance.

Data Fusion Maintenance: Services which employ local data fusion continuously seek new uninitialized SEs. Upon detection, these new SEs are initialized with data according to a well-defined, service-specific data fusion policy which takes into account the data of surrounding SEs in order to maintain the *continuity* of data properties. For example, in an area where the SEs of the SE infrastructure have been calibrated with position information, the Positioning Manager initializes newly distributed SEs with the calculated mean of all individual position coordinates obtained from SEs found in the vicinity.

Abstractions Maintenance: The *Location* abstraction is based on sets of SE identities. These sets have to be kept consistent when "old" SEs cease functioning and "new" SEs are redistributed over time. It is the task of the Location Manager to dynamically add newly detected SEs to the previously defined and recorded *Locations* in global or domain-wide Location Manager databases, as well as to store existing *Location* identifiers on new SEs in the respective *Locations*.

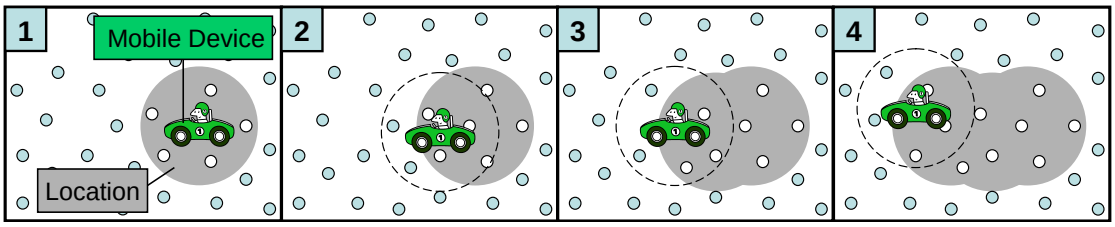

Figure 12.3.: Degeneration of a defined *Location* due to unsupervised maintenance

Pruning of Smart Entity Records: Some middleware services keep persistent records of the identities of individual SEs, such as the Location/Region Manager and Basic Map Manager service. In these cases it is important not only to incorporate the IDs of newly discovered SEs, but to regularly prune the affected databases in order to remove IDs that are no longer valid (e.g., due to defunct SEs). For instance, if a certain SE has not been detected during a well-defined number of subsequent *Location* or map updates, it can be marked as defunct and be removed from the respective databases.

Maintenance of Stateless Services. For stateless services which do not rely on information stored on SEs, the hot-integration of newly distributed SEs is trivial. Any locally detected SE can be utilized without prior initialization. In particular, no actions for maintaining the aforementioned fault-tolerance properties are required. With regard to our middleware, this is generally the case for all services which only require to read the identities of SEs, such as the Basic Map Manager service, for instance. Further examples are services that only store data transiently on the SEs, such as the Tracing and Tracking services, where the period of validity of the data is short compared to the expected lifetime of the SEs.

Measures Against Degeneration. A challenge of the hot-integration of new SEs is to restrict the area of activity per SDU. For instance, during infrastructure maintenance, it is non-trivial to determine which SEs belong to the current *Location* when new SEs are to be integrated by the Location Manager service, or across which area an SDU should be replicated by the Local Data Sharing service. Without special measures, a *Location*, for example, may degenerate through uncontrolled growth during maintenance (see Fig. 12.3).

To prevent this kind of degeneration, we have so far identified two strategies. Firstly, position information stored on SEs can be used to limit the expansion of the areas defining a *Location* during maintenance. For that, during the initial definition of a *Location*, the MoD uses the Position Manager to obtain the current position, which is stored along with the label of the *Location*. During maintenance, the MoD then uses its current position to decide whether a new SE is to be added to the set of SEs defining a particular *Location*. Similarly, replicated SDUs may be enriched with the original position information to restrict the area of replication. Secondly, by means of the Position Reckoning services, MoDs can determine their state of movement and progress, and use this information to maintain a safety distance between different physical places where maintenance tasks are performed.

12.5.6. Collaborative Bootstrapping and Maintenance

A super-distributed smart-entity infrastructure can be regarded as a scalable shared medium with virtually unlimited, independent, and highly distributed physical "access points", enabling MoDs to store and access data locally in the very location they occupy at the particular moment. This property makes the super-distributed smart-entity infrastructure particularly suitable to support collaborative scenarios where several independent physical entities work together on a single task in a highly decentralized and concurrent fashion (cf. Sections 11.1.2 and 12.5.6).

In our middleware for super-distributed smart-entity infrastructures we explicitly take advantage of this *aptitude for collaboration* among autonomous entities. In concrete terms, we exploit it for performing service initialization (bootstrapping) and maintenance tasks in a collaborative manner:

Collaborative Service Bootstrapping: Collaboration is supported for the bootstrapping of the following middleware services:

Location Manager: The Location Manager on a roaming MoD automatically defines new abstract *Locations* whenever the MoD reaches an area within the SE infrastructure that has not been processed before (Fig. 12.2(2)). This information is stored locally on the SEs and can be forwarded to a domain-wide service instance.

Position Manager: MoDs which have access to a third-party positioning service store obtained position information on nearby SEs (using the Position Manager) while moving in the SE infrastructure.

Map Manager: MoDs continuously chart areas of the SE infrastructure. This yields partial map views containing the positions of detected SEs or of identified *Locations*. The individual MoDs forward their observations to a global Map Manager instance residing in the network. There the partial maps are merged into a global map of the area, which is then made available for download to MoDs in return.

Collaborative Infrastructure Maintenance: All SE infrastructure maintenance tasks (including fault-tolerance maintenance) described above can be carried out in a collaborative fashion by the MoDs: each MoD performs the respective tasks at its current physical location, independently from the actions of MoDs in other places.

As we can see, the shared-medium characteristic of the SE infrastructure and its *locality* property enable the *self-organization* (e.g., partitioning the SE infrastructure into *Locations*) and *self-calibration* (updating positioning information stored in individual SEs) of middleware services over time. Taken together, the individual localized contributions of independent devices yield a collaborative effort on a global or infrastructure-wide scale.

12.5.7. Disconnected Operation

A major advantage of services based on a super-distributed smart-entity infrastructure is the capability of storing and retrieving data in situ by accessing the

physical SE memory (locality aspect). Consequently, for services that only require data stored on locally detected SEs (e.g., IDs of SEs, position coordinates, *Location* IDs, trace IDs, etc.), the MoD executing the services does not depend on network connectivity and remote services, but can operate in an autonomous and disconnected manner. Therefore, it will be unaffected by network outages or physical damage in other areas of the SE infrastructure.

Services that do not require constant remote access and whose local results obtained during offline operation can be unambiguously merged with the data maintained at a global service instance can continue performing their local tasks even in disconnected mode. Once connectivity is reestablished, results of local activities can be submitted to the global server instance for further processing.

In our specific case, after a period of disconnected operation, the Region Manager, the Map Manager, as well as the Directory services may reconnect to global service instances residing in the background infrastructure in order to submit newly defined *Regions* or created partial maps, or to update directory entries.

12.5.8. Data Consistency

In our approach, data consistency is an issue in the following situations:

Local Access to Physical SE Memory: The Hardware Abstraction Layer manages concurrent access to individual physical SEs. If that is technically unfeasible, it is important to detect read/write failures, which may be induced by interference or mobility and lead to inconsistent states of the physical SE memory. In our prototype, we use CRC checksums to detect inconsistencies during read/write operations. On error, failed read/write operations are repeated or the corrupted service data units are reset and reinitialized.

Remote Access to Virtual SE Memory: The virtual SE memory mirrors and extends the physical SE memory. Data consistency during remote access to the extended virtual memory is uncritical as the former can be protected using well-established synchronization mechanisms. Manipulative operations on the local physical SE memory require the update of the mirrored state of the SE in the virtual memory.

Remote Access to Global Service Instances: Data consistency is critical for services whose current operation depends on the results of previous activities carried out by other MoDs. In our approach, this only concerns the functioning of the Location Manager: it has to be prevented that multiple *Locations* with different identifiers are defined for the same physical place. As a solution, the Location Manager directly stores identifiers of newly defined *Locations* on the SEs at the physical place to inform other MoDs about previously performed *Location* definitions.

12.6. Prototypical Implementation Based on RFID Technology

To assess the practicability and effectiveness of our middleware architecture, we prototypically realized several exemplary services and applications. As enabling

technology for the construction of the prototypical SE infrastructure, we chose radio frequency identification (RFID). As a consequence, the SEs were represented by passive RFID tags, and the Hardware Layer in our prototype implementation constituted a super-distributed RFID tag infrastructure as described in Chapter 11. We therefore use the terms "RFID tag" or simply "tag" synonymously with "smart entity" in the remainder of this dissertation. The MoD executing the service middleware software was represented by a notebook computer, to which a mobile RFID reader and antenna were connected.

12.6.1. Hardware Layer and Hardware Abstraction Layer

As SEs we used ISO 15693 compliant Philips I·CODE (Type 1) transponders [Phi06] that operate at a frequency of 13.56 MHz. We distributed these tags over an L-shaped area of the floor space in a pseudo-random way. The RFID tags covered an area of approx. 6.3 m^2, with an average density of 39 tags/m^2.

Our realization of the Entity Read/Write service of the Hardware Abstraction Layer (Layer 1) is based on the RFIDStack [FL05b], which offers a manufacturer-independent programming interface to applications.

12.6.2. Exemplary Core Services

On top of the Entity Read/Write service, we implemented three essential services of the Core Services Layer (Layer 2): Local Data Sharing, Location Manager, and Position Manager. The implementation details are discussed in Sect. 13.3.

12.6.3. Higher-Level Services and Applications

Based on these basic services we also implemented and experimentally evaluated some higher-level middleware services and applications: services for tracing and tracking, self-positioning, and collaborative map-making.

The *Tracing and Tracking Service* features a tracing mode, which enables the MoD to leave a digital data trace, and a tracking mode, which allows a MoD to follow a previously laid trace. By using the Local Data Sharing service, trace data objects are stored in a redundant fashion on the RFID tags at the current position of the MoD at a well-defined update rate (specified in milliseconds). A more recent trace data object features a more recent timestamp. A MoD that follows a track (identified by a unique trace ID) continuously seeks tags containing trace data objects with the correct trace ID and the most recent timestamp.

The *Positioning Service* enables the MoD to store position information on or to retrieve it from individual RFID tags, either using the physical on-tag memory or a virtual tag database. The position calculation is performed with the assistance of the Position Manager service.

Based on three test runs at a speed of 50 cm/s for the MoD, the resulting mean absolute positioning error of the Positioning Service was approx. 15 cm for a density of 39 tags/m^2 of the test area. The maximum position calculation rate was up to 5 Hz using the virtual tag memory, and up to 2 Hz using the physical tag memory. A screenshot of an implemented graphical front end for the Positioning Service, which visualizes the positioning procedure, is displayed in Figure 12.4.

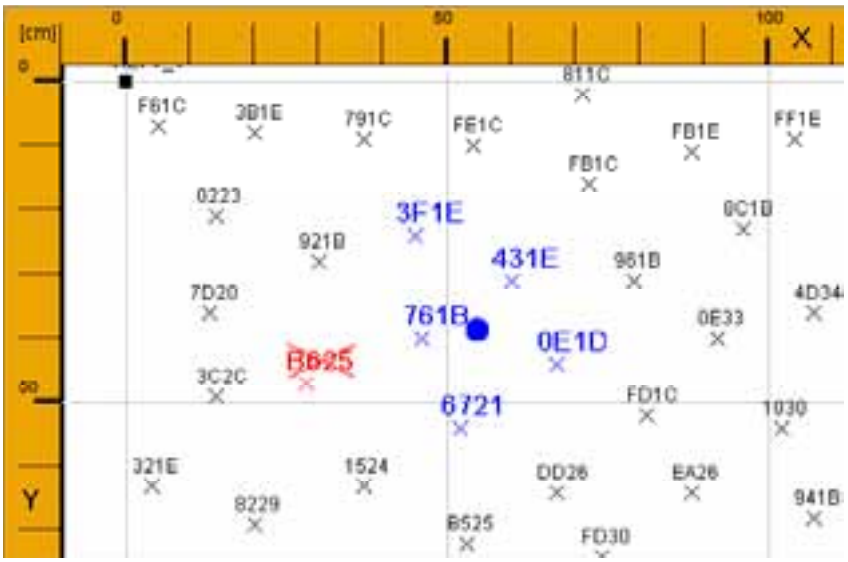

Figure 12.4.: Visualization of the positioning procedure during operation. The solid dot indicates the position of the MoD that was calculated from the position information obtained from the tags with the IDs `3F1E`, `431E`, `761B`, `0E1D`, `6721`. Tag `B625` had also been detected in the process but failed to respond during the position read attempt

We further implemented a prototypical *Collaborative Mapping* system. It provides a rotation-sensor-assisted map-making service to independent, autonomous vehicles that carry a MoD, enabling them to perform the *localization* and *mapping* of RFID tags within the test area. The Collaborative Mapping system further provides a service for the *merging* of partial mappings that were constructed independently by different vehicles as part of a collaborative effort.

In Chapter 13 we give a more detailed description and an experimental evaluation of the developed prototypical reference implementations.

12.7. Summary

We described how *super-distribution of smart entities* can serve as a design principle for the realization of fault-tolerant location-dependent services. We outlined a middleware which, by means of a portfolio of location-dependent services, facilitates the development of dependable location-aware applications for mobile devices based on super-distributed smart entity infrastructures. We further presented a number of general middleware design challenges for such infrastructures, and described concrete solutions for fault-tolerance management, infrastructure maintenance and bootstrapping, and enforcing data consistency.

In the following chapter, we demonstrate the applicability and feasibility of our approach. We do so by providing the reader with a detailed description and analysis of several prototypical reference implementations based on radio frequency identification (RFID) as an enabling technology.

12.8. Related Work

Our concept of deploying small computational entities on a large scale on a physical substrate is related to the idea of mixing computer particles with "bulk materials", which Abelson et al. called *amorphous* or cellular computing [AAC+00]. The primary goal of amorphous computing is to obtain a "collection of computational particles" that achieve a "cellular cooperation" as found in biological organisms.

In contrast to this visionary concept, we more concretely aim at doting a classical physical substrate with smart entities, such as a floor space equipped with a dense distribution of passive RFID tags, which allows mobile devices to interact *in situ* with their physical environment at their respective place. Fundamentally, our approach is a generalization of the concept of *super-distributed RFID tag infrastructures*, which we introduced in Chapter 11.

A dense and redundant distribution of tiny objects is also a common characteristic of *wireless sensor networks*, which consist of a large number of small, autonomous sensor nodes. Sensor networks are concerned with the observation and monitoring of real-world phenomena within a certain environment, typically based on the collaborative effort of a large number of nodes [ASSC02b]. For achieving fault-tolerance in the face of hardware failures, sensor nodes are typically distributed in a redundant manner. In contrast to sensor networks, our work is concerned with the deliberate augmentation of the physical environment with the explicit goal of enabling individual mobile devices (and their users) to directly and locally interact with SEs placed in the physical environment. Communication between the SEs themselves is much less an issue than in sensor networks.

There have been various projects investigating middleware support for location-aware services and applications based on wireless networks and on remote services provided by background computing infrastructures. Two examples for modular location-aware service and application platforms that are represented by distributed processing infrastructures based on the CORBA middleware platform as enabling technology are LASAP [PPZ99] and MiddleWhere [RAMC+04]. An example of a middleware for location-based services that delivers location-aware content to mobile users on the basis of Web services and the positioning capability of a wireless network infrastructure is LORE [CCR+04]. In contrast, our middleware approach specializes in providing location-aware services to mobile devices in a self-sufficient manner by relying on physical spaces doted with computerized entities for storing and retrieving information in situ.

Acknowledgments

The author wishes to acknowledge Vito Piraino for his help with the requirements analysis of existing location-aware ubiquitous computing systems, and for his work on the implementation of selected middleware service [Pir04].

13. Prototypical Implementation of a Fault-Tolerant Location-Aware Service Middleware Based on Super-Distributed RFID Tags

In this chapter we provide evidence of the feasibility and effectiveness of the middleware architecture for mobile devices presented in Chapter 12, which employs dense distributions of small computerized entities for providing fault-tolerant location-aware services. We do so by describing exemplary implementations based on radio frequency identification (RFID) as an enabling technology. Firstly, we present prototypical implementations of the hardware abstraction layer and of selected core middleware services. The latter enable a mobile device to store and retrieve data and position information in physical places in a fault-tolerant manner, and to identify places based on a location abstraction which is robust against failure of individual tags. Secondly, we investigate the feasibility of some higher-level services and applications by developing and evaluating prototypical systems for tracing and tracking, self-positioning, and collaborative map-making.

13.1. Motivation and Background

In Chapter 11, we showed how – different from conventional means of RFID tag deployment and utilization – massively-redundant tag distributions provide novel RFID-based services and applications to mobile devices (MoDs). By deploying cheap passive RFID tags (i.e., tags without a built-in power supply) in large quantities and in a highly redundant fashion over large areas or object surfaces, we obtained a super-distributed RFID tag infrastructure (SDRI). Based on such an SDRI, we identified a number of technical challenges and described potential benefits and first prototypical results. The practical relevance of this concept is reflected in the recent appearance of industrial products that make use of such redundant RFID tag distributions, such as the "first carpet containing integrated RFID technology" presented by Vorwerk in cooperation with Infineon Technologies [Vor05].

As a generalization of the SDRI concept, we then proposed *super-distribution* of small computerized (and therefore "smart") entities as a general design principle for the development of reliable and highly available location-dependent services for MoDs. For that, we developed a layered *service middleware architecture*, which we presented in Chapter 12. This middleware architecture exploits two fundamental characteristics of the resulting infrastructure for achieving fault tolerance and ser-

viceability: the high degree of redundancy with regard to smart entities (abundance aspect), and the support for localized interaction between mobile devices and their immediate physical environment (locality aspect).

Earlier we focused on *theoretical* middleware aspects. In this chapter, we describe a number of concrete prototypical implementations based on super-distributed smart entities, using RFID as an enabling technology. In doing so, our major aim is to provide first-hand evidence of the *practicability* and *effectiveness* of the suggested approach by demonstrating the capabilities and performance of exemplary middleware service implementations, rather than presenting specific state-of-the-art solutions for the particular application domains we cover in the process.

In the following sections, we adhere to the terminology proposed above in Chapter 12.2.1, where a *smart entity* (SE) is defined as a physical artifact that is enhanced by embedded computing technology in such a way that it has a globally unique identifier, a built-in memory with data read/write capabilities, and support for close-range wireless ad hoc communication. Likewise, we refer to *super-distribution of smart entities* as the process of deploying and distributing SEs in a dense, highly redundant fashion. The resulting substrate is called a *super-distributed smart-entity infrastructure*.

13.2. Overview of Middleware Implementation

Our service middleware for super-distributed smart-entity infrastructures described earlier in Chapter 12 is based on a five-layered architecture (see Fig. 12.1): The distributed physical smart entities in their entirety constitute the physical infrastructure on the lowest level (*Hardware Layer* or Layer 0). The access to this layer is controlled by the *Hardware Abstraction Layer* on the next higher level (Layer 1). It is represented by an *Entity Read/Write* (ERW) service, which defines a generic and unifying interface to the underlying physical SE infrastructure. The *Core Service Layer* (Layer 2) consists of fundamental abstractions and generic services that operate with individual SEs by means of the ERW service. The *Higher-Level Service Layer* (Layer 3) is represented by a collection of specialized services and service templates. These services do not directly operate on individual SEs of the underlying physical infrastructure but rely on the core services instead. Finally, in the *Application Service Layer* (Layer 4), we find application-specific services and specialized instantiations of service templates.

For our prototypical implementation, we selected a number of exemplary middleware services based on both a bottom up and top down approach. On the one hand, we implemented services of the lower layers that provide general basic functionality, which includes the Hardware Layer, the Hardware Abstraction Layer, and three essential services of the Core Services Layer: Local Data Sharing, Location Manager, and Position Manager. On the other hand, based on these services, we investigated the feasibility and practicability of some higher-level services by developing and evaluating prototypical systems for tracing and tracking, self-positioning, and collaborative map-making.

In our implementations, the SEs were represented by passive RFID tags. Therefore, in accordance to the terminology used in Sect. 12.6, we use the terms "RFID tag" or simply "tag" synonymously with "smart entity". Further, unless stated otherwise, the MoD executing the service middleware software was represented by a

notebook computer, to which a mobile RFID reader and antenna were attached to enable a localized interaction with the SDRI.

13.3. Basic Middleware Services

We prototypically implemented the Hardware Layer consisting of super-distributed RFID tags, the Hardware Abstraction Layer represented by the Entity Read/Write service, and three essential services of the Core Service Layer: Local Data Sharing, Location Manager, and Position Manager.

13.3.1. Hardware Layer: Super-Distributed RFID Tag Infrastructure Prototype

The RFID hardware we used for the SDRI consisted of ISO 15693 compliant smart labels (transponders) that operated at a frequency of 13.56 MHz. As transponders, we employed Philips I·CODE tags (Type 1) [Phi06], with a dimension of 7.5 × 4.5 × 0.1 cm. The I·CODE RFID tags feature 64 byte of physical memory, which is organized into 16 slots á 4 byte (of which 11 slots are rewritable). This allowed us to store the data of several middleware services (e.g., Position Manager and Tracing and Tracking Service) directly on the physical memory of individual tags during our experiments.

Dimension of plastic foil templates	123 × 128 cm
Mean distance between two adjacent RFID tags	17.5 cm
Standard deviation of tag distribution	2.1 cm
Number of tags per plastic foil template	61 tags/foil
Average area covered by a single RFID tag	258 cm^2
Average RFID tag density per square meter	39 tags/m^2

Table 13.1.: Properties of plastic foil templates used for building a prototypical super-distributed RFID tag infrastructure (SDRI)

For building the SDRI, we attached the transponders onto four identical plastic foils using the same pseudo-random distribution pattern. This yielded four RFID-tagged templates with equal characteristics as given in Table 13.1.

13.3.2. Hardware Abstraction Layer: Entity Read/Write

For the realization of the Entity Read/Write (ERW) service on the Hardware Abstraction Layer, we used the RFIDStack [FL05b], which offers a manufacturer-independent API to applications and incorporates drivers for various types of RFID hardware. Based on the RFIDStack, the ERW service provides the interface for writing data to and reading data from the underlying RFID tags of the SDRI, masking the complexity and hardware-specific characteristics of the underlying RFID hardware from the higher service layers. The writing of data can either be performed physically by writing to the physical tag memory, or virtually by storing the data in the so-called virtual tag memory. The latter is managed by a service

instance of the RFIDStack residing in the Internet, which can be accessed by means of XML messages sent via a TCP connection [FL05b]. The virtual tag memory not only mirrors the physical memory of a tag, but also provides an extended storage space. Our ERW implementation only allows a MoD to access the virtual memory of a tag if that entity is physically present within communication range.

The ERW service also implements the data management for the physical and virtual tag memory. It emulates a simple file system for the physical tag memory, where Service Data Units represent files and the Smart Entity Directory represents the root directory. A Service Data Unit constitutes a service-specific data unit that encapsulates the information a service requires to be stored on a single tag for a well-defined purpose. To detect corrupted data units on tags caused by interrupted, incomplete write operations, CRC error checking is performed.

In particular, the ERW service provides the following basic methods for accessing the physical memory of individual tags: `listTags`, `listTagDirectory`, `writeTagFile`, `readTagFile`, `deleteTagFile`. Parameters include tag ID, file type, file data, and flags that indicate the use of the virtual memory and declare if a file should be stored persistently or can be overwritten at a later point in time (persistence flag).

13.3.3. Core Service Layer: Local Data Sharing

The Local Data Sharing (LDS) service provides MoDs with an API for sharing data with other devices in physical places of the SDRI. In doing so, the LDS service exploits the high tag density in the SDRI for fault-tolerant data storage by replicating Service Data Units across multiple tags in antenna range at the current location of a MoD. Data can be shared in situ by using method `shareData`, which is parameterized with the service-specific data type and the persistence flag. Previously shared data can be retrieved by means of the `getData` method.

The API of the LDS service allows the user to set the replication degree as an absolute number or as a relative percentage value. These settings apply to the initial replication and the later replication maintenance procedure. The replication itself is performed on a best effort basis and its management is hidden from the service clients. For accessing the tags of the SDRI, the LDS service uses the API of the ERW service. The LDS service further allows to set a tolerance threshold for the number of failed tag identify/read/write attempts of the underlying ERW service. For example, if data is to be read from / written to eight different tags, failed read/write attempts for two of the tags are tolerated given a tolerance threshold of 25%. This enables the service to deal with known imperfections of RFID systems (e.g., tags in range may not be detected even if physically present, or read/write operations may abort [FL04]). When Service Data Units are retrieved from local tags, the LDS service transparently filters duplicates.

13.3.4. Core Service Layer: Location Manager

The Location Manager (LM) service provides an API to define and resolve abstract locations. A *Location* has a unique identifier and is defined by the set of (stationary) SEs that are detected in a well-defined range of the MoD (i.e., in the range of the RFID antenna of the MoD) executing the service. The *Location* identifiers are directly stored on the defining SEs themselves.

The main contribution of the LM is the `getLocation` method, which determines an abstract *Location* L as the set of RFID tags $tagIDSet_l$ detected at the respective physical place l in the SDRI: $L := tagIDSet_l := \{tagID_t : inRange(t, l, r)\}$, where $tagID_t$ is the unique identifier of tag t, and $inRange(t, l, r)$ a Boolean predicate that equals `true` iff tag t is within range r of the field of the RFID antenna at place l and `false` otherwise. In our prototype system, the range r of the RFID system was defined by the characteristics of the used RFID hardware. Ideally, the range of the RFID reader/antenna should be customizable to enable the integration of different RFID systems with variable characteristics.

If the `getLocation` method is called to determine the *Location* of the current place, then the LM service searches for predefined *Location* identifiers on all tags in range r of the antenna (i.e., of the MoD). The *Location* whose identifier is stored on the majority of the detected tags is returned as the current *Location*. In case no predefined *Location* is available, or if the number of tags containing the dominant *Location* identifier is below a well-defined percentage value T, then the LM automatically defines a new *Location* and stores the corresponding *Location* identifier on the affected tags. This ensures that adjacent *Locations* differ in at least $(100 - T)\%$ of the tags, which in return enables a robust and selective *Location* detection in situations where individual tags fail to respond temporarily.

13.3.5. Core Service Layer: Position Manager

The main contributions of the Position Manager service are the methods `getPosition` and `setPosition`. The `setPosition` method enables a MoD to locally store its current position p^M obtained from a third party positioning service on the physically proximate tags. In doing so, for each tag, the new position p^t_{new} is calculated as the weighted mean of the position p^M of the MoD and the old position p^t_{old} of the tag, using the number of previous write operations w as a weight: $p^t_{new} := (p^M + w \cdot p^t_{old}) : (w + 1)$. Vice versa, upon calling `getPosition`, the Position Manager first scans all tags in antenna range and extracts their individual position coordinates if available. Then it calculates an estimate for the current position as the arithmetic mean over all obtained individual tag positions.

13.4. SDRI Tracking and Positioning Prototype

The SDRI Tracking and Positioning prototype provides two main services: laying and following of data traces, and self-positioning.

13.4.1. Prototype Description

We developed a fully functional SDRI Tracking and Positioning prototype, which consists of two major hardware components. Firstly, a trolley with the RFID equipment (RFID reader and antenna) and the MoD (in our case represented by a notebook computer running the SDRI Tracking and Positioning application). Secondly, four RFID-tagged templates forming a prototypical SDRI (Fig. 13.1).

The RFID hardware consisted of an ISO 15693 compliant mid range RFID reader,

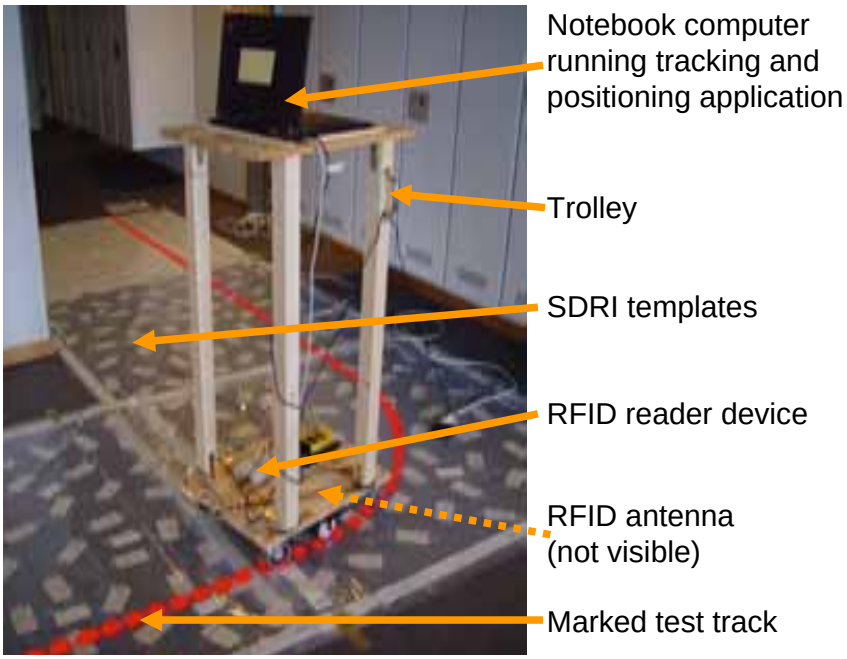

Figure 13.1.: Measurement trolley and prepared test track

and an external mid range RFID antenna.[1] The RFID reader supported collision resolution, which enabled it to simultaneously identify multiple transponders within antenna range. The RFID antenna was attached underneath the center of the bottom pane of the trolley, at 10 cm above the floor space. At this distance, the approximately square operating area of the RFID antenna was about 50 × 50 cm. For constructing the prototypical SDRI, the four RFID-tagged templates described earlier in Sect. 13.3.1 were arranged in an L-shape around a corner of a corridor in our office building (Fig. 13.1). On the templates, we manually marked a test track for our experiments with a total length of 526 cm.

13.4.2. SDRI Tracing and Tracking Service

The SDRI Tracing and Tracking Service features a tracing mode, which enables the MoD to leave a digital data trace in the SDRI, and a tracking mode, which allows a MoD to follow a previously laid data trace. Each mode itself is divided into a basic and advanced version, which we describe in the following.

Tracing Mode

A *basic trace* is represented by a sequence of trace data objects (TDOs) stored on tags of the SDRI. Each *trace data object* consists of an anonymous trace identifier (trace ID), which is generated by random, and a timestamp. A trace ID has only to be unique in the local area where it is applied, but not on a global scale. Further, all TDOs are flagged as non-persistent, and over time, the SDRI Tracing and Tracking Service overwrites the oldest TDOs on a tag with newer traces if the MoD is running low of memory space.

[1] Manufacturer: Feig Electronic, model: OBID *i-scan* reader HF ISC.MR100 and OBID *i-mid* antenna ISC.ANT340/240

In our prototypical implementation, we replaced the timestamp in the TDO with a trace counter serving as logical clock to obtain a more compact, memory-space-saving representation. This was feasible since it is usually only necessary to locally distinguish the age of detected TDOs belonging to the same trace, which we achieve by applying a sliding-window approach. In addition, we adapted the TDO overwrite strategy to selecting a random TDO for replacement, as the use of logical clocks no longer allows to identify the oldest TDO on a tag. Memory-wise, we used 1 byte for the trace ID and one for the trace counter (with a window size of 12) per basic TDO, which fit into a single slot of our physical RFID tag memory.

If tracing is active, new TDOs are stored in a redundant fashion on the RFID tags at the current position of the MoD (by using the Local Data Sharing service) at a well-defined update rate (specified in milliseconds). For preventing repetitive trace updates at the same physical location, which would lead to a discontinuity of the trace counter values, the IDs of the locally detected tags are cached. A new TDO is only written to the SDRI if at least K percent of the local tag IDs have changed. Concretely, we used a trace update rate of 500 ms and set the update tolerance to $K = 50\%$.

The *advanced tracing mode* uses position information (e.g., obtained from the Position Manager or from a third-party positioning service) to create an *augmented trace*: the individual trace data objects are augmented with the current information about direction (orientation), change of direction, and speed of the MoD.

Tracking Mode

The tracking mode of the SDRI Tracing and Tracking Service enables a MoD, the *follower*, to follow a trace by detecting the corresponding TDOs in the tags of the SDRI. We call the MoD that previously laid the trace the *forerunner*. Initially, the forerunner has to reveal its randomly chosen trace ID of the trace to the devices that are to become its followers, and to inform them about potential starting points for picking up the trace (which are not necessarily equal to the starting point of the trace).

Once a follower has detected or rediscovered the trace (i.e., tags in the SDRI which contain a TDO with the forerunner's trace ID), the follower repeatedly searches for tags with more recent trace information and moves into this direction. More precisely, the follower continuously seeks TDOs of the wanted trace ID with either a more recent timestamp, or with a higher trace counter value (based on the counter window calculated using modular arithmetic). In our system, the detected trace counter values for a specified trace are displayed in a graphical user interface window (GUI). If an RFID tag map of the prototypical SDRI is available, the GUI visualizes the tags of the trace that have been detected so far, and highlights the most recent trace information. In case of an augmented trace, the GUI also displays the augmented information, such as the current direction and change of direction (as numerical values and visually by means of an arrow symbol).

13.4.3. SDRI Positioning Service

The SDRI Positioning Service enables the MoD to store position information to or to retrieve it from individual RFID tags of an SDRI, either using the physical on-tag memory or a (local or remote) virtual tag database.

Calibration Mode

For the calibration of the SDRI with position information, the SDRI Positioning Service supports two modes of operation: Firstly, the *exact* calibration mode allows the user to calculate the individual tag positions of all RFID tags of an SDRI template at once, based on two manually entered reference positions per template. The determined exact tag positions are then stored on the physical tags and/or in the virtual tag database. The physical tag calibration procedure is supported by a tool that shows the progress and status of the calibration with the help of a graphical display.

Secondly, the *incremental* calibration of the SDRI uses the position information of a third-party positioning service to update the position coordinates on the individual tags by calculating a new weighted mean as described in Sect. 13.3.5. This procedure can be performed in a collaborative fashion by independent MoDs. In the process, the accuracy of individual tag position coordinates usually increases with the number of positions that are stored on the respective tags. As the actual positions of the MoDs performing the calibration are typically scattered around individual tags, the errors of the single position values that are averaged tend to cancel each other out.

Positioning Mode

The implemented *position calculation* or *positioning* procedure of the SDRI Positioning Service is based on the positioning procedure of the Position Manager. First the tag position coordinates stored on the single RFID tags within antenna range are retrieved. Then the arithmetical mean of the obtained single tag position coordinates is calculated and used as the estimated position (x, y, z) of the MoD.

13.4.4. Experimental Results

We performed our experiments by pushing the trolley at a constant speed along the marked test track (Fig. 13.1). We further calibrated the tags of the SDRI with local positioning coordinates using the exact calibration tool.

Efficiency of Virtual and Physical Tag Memory Access

For our positioning measurements, we used the virtual and physical tag memory.

For accessing the virtual tag memory, which was maintained in a database on the MoD itself, it was sufficient for the ERW service to retrieve the IDs of all RFID tags within antenna range with a single command call (`identify`). The duration of the `identify` command was independent from the number of tags within range, and took approximately 200 ms on average (using 16 time-slots for multi-tag-detection as part of the anti-collision protocol of the reader). This enabled a maximum rate of up to 5 Hz for multi-tag detection and subsequent position calculation.

The efficiency of the physical tag memory was more than one order of magnitude lower, since our particular RFID hardware required sequential scans for reading out a data slot from multiple tags: one `identify` command followed by a separate `read` command for each detected tag. In our implementation, we needed two physical memory slots to store positioning coordinates on a tag. Therefore, for

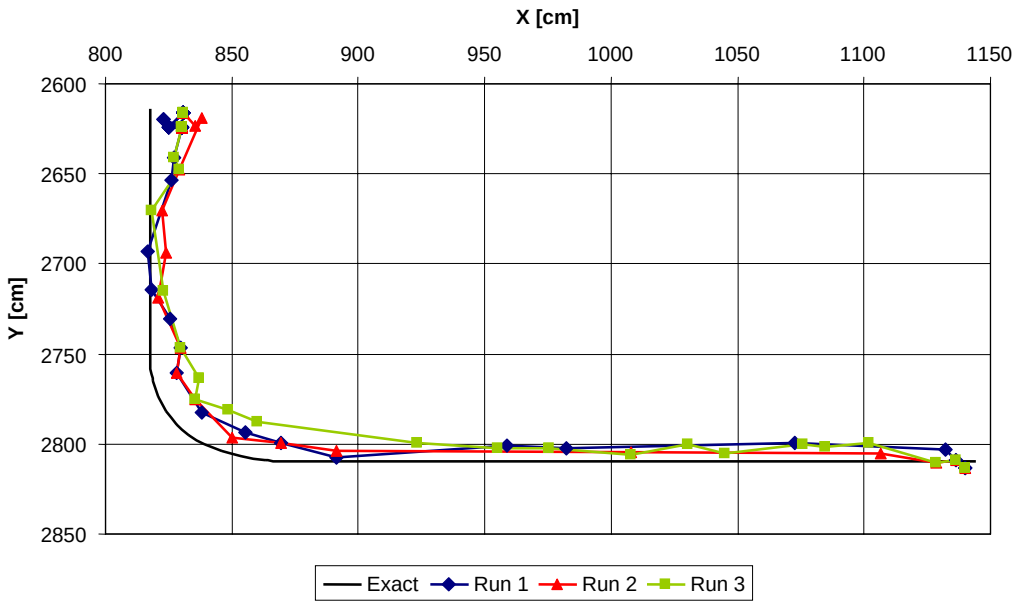

Figure 13.2.: Three positioning experiments of the SDRI Positioning Service performed at 50 cm/s

attempting to read the two data slots from four RFID tags detected during an inquiry, the duration of the scan varied from approximately 2 seconds (8 successful reads), if no errors occurred, to up to 5 seconds (8 failed reads) in the worst case if all eight sequential read operations failed. These numbers are based on timing measurements for successful and failed attempts for reading a single data slot, which for our RFID hardware were approx. 250 ms and 600 ms, respectively. However, by using a more advanced RFID system that supports the direct and parallel reading of a data slot from multiple tags in range without a prior `identify` operation, the duration of the physical tag memory access can be reduced to the order of magnitude of the duration of the virtual tag memory access.

Accuracy of the Positioning Procedure

Due to the comparably slow physical tag memory access of our RFID hardware, we used the virtual tag memory for our experiments. We performed three test runs at a speed of 50 cm/s, using exact manual measurements of the test track as reference (Fig. 13.2). The resulting mean absolute positioning error was approx. 15 cm. Given our specific configuration, the maximum tolerable speed of the trolley is 2.5 m/s, which is determined by the tag inquiry time of approx. 200 ms (required by the ERW service for determining the tag IDs for accessing the virtual tag memory) and the length of the antenna field in moving direction of 50 cm.

13.5. Collaborative SDRI Mapping Prototype

The prototypical Collaborative SDRI Mapping system has two main tasks: The *localization* and *mapping* of RFID tags in an SDRI by means of autonomous vehicles, and the *merging* of overlapping partial RFID tag mappings, which were constructed independently from each other by these vehicles as part of a collaborative effort.

We do not aspire to contend with state-of-the-art solutions for the general collaborative map-making problem, which has been in the focus of research in the domain of mobile robots for decades (cf. the work by Burgard, Fox, et al. [BMF+00, FBKT00], for instance). Our primary goal is to demonstrate the feasibility and practicability of using a super-distributed RFID tag infrastructure for the realization of collaborative activities, which is not considered by traditional map-making systems. In contrast to our approach, RFID tags for positioning have so far only been used in the function of dedicated artificial *landmarks* on walls or floor spaces, providing auxiliary support to dedicated positioning and navigation systems [KBPD97, NLLP03, HBF+04].

13.5.1. Prototype Description

The Collaborative SDRI Mapping prototype consists of the following components: (1) a model vehicle, (2) a prototypical SDRI, (3) an on-board vehicle control application (for evasive driving and dead reckoning), (4) an off-board RFID tag mapping application, and (5) a stand-alone collaborative map-merging application for fusing partial map observations obtained during independent test runs.

The *model vehicle* was constructed using Lego Mindstorms [LEG06] technology. It is self-propelled, featuring two actuated parallel wheels in the back (each equipped with a rotation sensor and an electrically powered motor) and one castor wheel in the front for stabilization. A bumper sensor connected to a front bumper is used for collision detection. An on-board LEGO Mindstorms RCX controller hosts the software for controlling the motors of the vehicle, and for monitoring the rotation and bumper sensors. In addition, the model vehicle is equipped with an on-board RFID reader (Fig. 13.3) and an RFID antenna[2] mounted at the bottom at 1 cm distance from the floor space (Fig. 13.4). Due to the size of the model vehicle, the vehicle control application was executed on a separate notebook computer, which was connected to the RCX controller and the RFID reader by cable.

For obtaining a *prototypical SDRI test area*, we evenly distributed 32 mu-chip inlets across a wooden panel of the size of 50 × 50 cm (Fig. 13.3). This corresponds to a tag density of 128 tags/m^2. Each mu-chip tag features a unique 128-bit ID stored in its read-only memory (ROM). The test area of was rounded off with a solid wooden barrier to mark off its boundaries.

The *on-board vehicle control application* is executed on the RCX controller and performs the following actions: It triggers an evasion manoeuvre whenever the bumper sensor connected to the front bumpers reports an obstacle. The RCX control application also continuously monitors the two rotation sensors and calculates the current position by means of a basic dead reckoning algorithm. The *RFID tag mapping application* is executed off-board on the notebook computer. The application is connected to the RFID reader and continuously maps detected RFID tags, using the latest dead reckoning position information obtained from the RCX controller of the model vehicle as reference position.

The *collaborative map-merging application* merges overlapping partial map observations, which were created during independent map making runs, with a single, gradually growing comprehensive map of the area. The map merging algorithm uses

[2]Manufacturer: Hitachi Kokusai Electric Inc., model: reader MRE200 No. 1010 and antenna PA1-2450AS

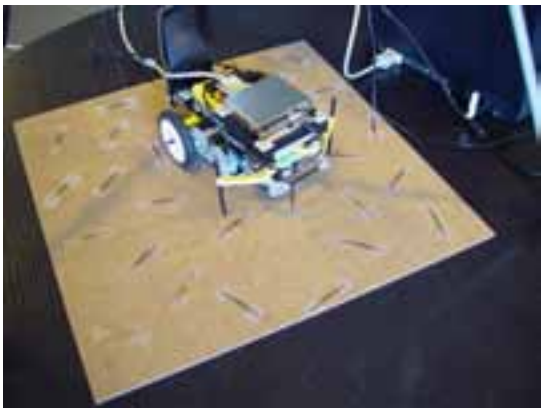

Figure 13.3.: Model vehicle with mu-chip reader on top of the Lego RCX, within the prototypical SDRI tagged with mu-chip RFID inlets

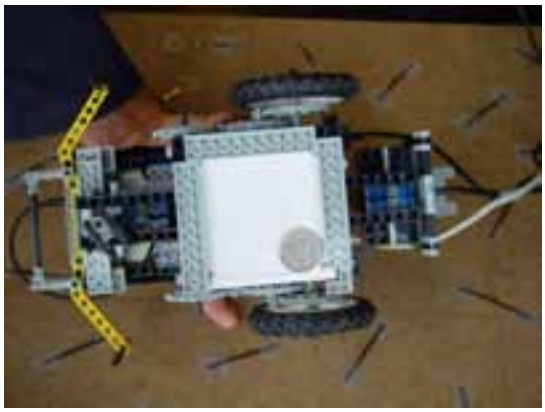

Figure 13.4.: Bottom view of the model vehicle prototype showing the wheel configuration, front bumper, and the mu-chip antenna

an affine coordinate transformation between two arbitrary maps with different local (or global) coordinate systems. The transformation is unambiguously defined by a translation vector and a rotation angle given two or more overlapping tags (i.e., tags that are contained in both maps). The affine transformation is calculated numerically using a least squares metric for minimizing the overall transformation error.

13.5.2. Experimental Results

Experimental Method and Validation

Four map-making test runs were carried out in our test area of 25 dm^2. Starting from a random position (which served as the origin of the local coordinate system for the measurement), the model vehicle drove along a straight trajectory within the SDRI at a constant speed of 3.6 cm/s. Whenever the vehicle's bumpers hit the encircling barriers, the vehicle stopped and performed an approx. 90-degrees turn on the spot, and resumed its straight movement. While driving, the off-board application recorded the tag IDs together with the corresponding local position coordinates of the tags detected by the RFID reader on the vehicle. The position coordinates were obtained from the dead reckoning program running on the vehicle's RCX controller. Each test run lasted approx. 90 seconds, during which the vehicle performed 6 turns (each of which took approx. 6 seconds). Thus, on average, the vehicle covered a distance of approx. 200 cm per test run.

To validate our experimental results, we have manually measured the exact local position coordinates of all RFID tags in the test area as a reference. To assess the quality of an experimental RFID tag map, we calculated the overall minimum, maximum, and mean absolute tag localization error. For an individual tag, the localization error was determined by calculating the Euclidean distance between its estimated position and its corresponding exact reference position.

Dead Reckoning Error

The driving distance of the model vehicle was approx. 0.33 cm per rotation sensor increment. The average absolute error of the dead reckoning algorithm for an approx. 90° turn of the vehicle on the spot was about 4%, and its lateral drift approx. ± 7 cm per meter during straight driving. When considering several consecutive turns, the occurring negative and positive errors partly annihilate each other, leading to a lower effective error. In our case, the overall error of six consecutive turns was reduced to approx. 1.4 %, which corresponds to an accumulated drift of only about 2 cm per meter of straight driving.

Tag Localization Error

The specific RFID antenna we used detected tags inside an area of approximately 6 *times* 9 cm around its center point, at 1 cm distance from the floor space. Since each mu-chip of our SDRI test area covered an area of approx. 78 cm^2, only one tag was within antenna range at a time. Therefore, whenever the model vehicle took its current reckoned position as a position estimate for a detected RFID tag, the error caused by the uncertainty about the exact tag position within the antenna field (i.e., tag reception area), which we refer to as *tag localization error*, added to the dead reckoning error.

In our prototype system, the tag localization error equaled the distance between the center of the tag reception area of the antenna and the center point of the mu-chip inlet. Concretely, given that the center point of the vehicle is also the center point of the RFID antenna tag reception area, the mean tag localization error amounted to approx. 2.7 cm. In the worst case, if a detected tag was situated in one of the corners of the tag reception area, the resulting maximum tag localization error was approx. 5.4 cm.

Tag Mapping Error

During the mapping, the deviation e_{TP} of experimentally measured tag position coordinates from the true coordinates, which we call *tag mapping error*, is determined by two factors: the error e_{DR} of the dead reckoning system (which is proportional to the distance traveled since the initial starting position was set), and the tag localization error e_{TL}, which depends on the properties of the RFID hardware and RFID tag distribution: $e_{TP} = e_{DR} + e_{TL}$.

Evaluation of Mapping Procedure

As a result of the four map-making test runs, four partial maps were created. In the process, on average 11 tags were detected per test run, and 21 different tags were detected altogether. Each two created maps overlapped in two or more tags. The resulting tag mapping errors for the tags of each partial map in comparison to the tags of the exact reference map are shown in Table 13.2. The average tag mapping error over four experiments was 4.1 cm, with little variation (standard deviation σ = 1.4 cm). The overall maximum tag mapping error remained below 8 cm.

Partial map	#Tags	Min. error	Max. error	Mean error	Std. dev. of mean error
1	10	2.5 cm	6.7 cm	4.3 cm	1.3 cm
2	9	1.0 cm	5.2 cm	3.2 cm	1.3 cm
3	11	1.9 cm	7.3 cm	4.3 cm	1.8 cm
4	14	2.0 cm	7.9 cm	4.4 cm	1.3 cm
Average:	11	1.9 cm	6.8 cm	4.1 cm	1.4 cm

Table 13.2.: Tag mapping errors of four experimentally constructed partial maps

Evaluation of Map Merging Procedure

To assess the robustness of our map merging procedure with regard to the order in which overlapping maps are merged, we have joined the four partial maps in different sequential orders and compared the resulting minimum, mean, and maximum tag mapping errors.

Merged maps	#Tags	Min. error	Max. error	Mean error	Std. dev. of mean error
1+2	15	1.5 cm	8.1 cm	3.9 cm	1.5 cm
1+3	15	1.4 cm	9.2 cm	5.2 cm	2.5 cm
1+4	21	1.0 cm	10.0 cm	4.7 cm	2.4 cm
2+3	17	1.0 cm	7.1 cm	4.0 cm	1.8 cm
2+4	16	1.2 cm	7.9 cm	4.1 cm	1.6 cm
3+4	18	1.3 cm	8.0 cm	4.2 cm	1.9 cm
Average:	17	1.2 cm	8.4 cm	4.4 cm	2.0 cm

Table 13.3.: Tag mapping errors of pairwise merged partial maps

In a first step, we merged the individual maps pairwise. The results show a slight increase of the mean tag mapping error to 4.4 cm, with a higher variability (σ = 2.0 cm), as shown in Table 13.3. The mean absolute tag mapping error increased slightly to 8.4 cm, with a new overall maximum error of 10.0 cm. The results differ significantly for each pairing of partial maps. An explanation for this observation is that – at this stage – a better map merging result can be expected for maps that have more tags in common.

Merged maps	#Tags	Min. error	Max. error	Mean error	Std. dev. of mean error
(1+2)+(3+4)	21	0.5 cm	7.6 cm	3.8 cm	1.8 cm
(1+3)+(2+4)	21	1.6 cm	7.6 cm	4.2 cm	1.6 cm
(1+4)+(2+3)	21	0.8 cm	7.7 cm	3.9 cm	1.8 cm
Average:	21	1.0 cm	7.6 cm	4.0 cm	1.7 cm

Table 13.4.: Tag mapping errors of maps obtained after two consecutive merging operations

In a second step, we merged the previously paired maps. The resulting errors are shown in Table 13.4. We can see that the mean tag mapping error stabilized

at 4.0 cm, with a lower standard deviation than in the case of the original partial maps. A stabilization can also be observed with respect to the minimum and maximum errors. The maximum tag mapping error after two consecutive map merging operations has even dropped below the initial values to 7.7 cm. Apparently, independently from the merging order, the errors with opposite signs tend to partially cancel each other out as the estimated tag positions of all available partial maps are eventually combined.

13.6. Conclusion

Based on the service middleware architecture for super-distributed smart-entity infrastructures described in Chapter 12, we prototypically implemented basic middleware layers and services with the help of RFID technology: the Hardware Layer, the Hardware Abstraction Layer, and the three essential core services Local Data Sharing, Location Manager, and Position Manager. We demonstrated the application of these services by developing and evaluating systems for tracing and tracking, positioning, and collaborative map-making.

The SDRI-based tracking and positioning system was implemented on top of two core middleware services, which rendered it fault-tolerant with respect to individual tag failures: (1) the tracking and positioning system redundantly stores trace data objects in physical places using the Local Data Sharing service, and (2) it exploits the data fusion capabilities of the Position Manager, which allows the service to tolerate the unavailability of single tags by interpolating the position coordinates of the MoD at a physical location. By means of experimental evaluation we demonstrated that our positioning service provides an average accuracy of approx. ± 15 cm at walking speed in our prototypical SDRI with a tag density of 39 tags/m^2. We consider this a promising result and a strong indication for the practicability and effectiveness of our approach, in particular considering that we used off-the-shelf RFID equipment that was not optimized for use in mobile environments.

The prototype system for the collaborative mapping of super-distributed smart entity infrastructures used mu-chip RFID tags as smart entities and low-cost rotation sensors for implementing the dead reckoning system. We experimentally evaluated an application for merging partial SDRI mappings created independently by autonomous MoDs. We observed that the mean tag mapping error stabilized on the level of the corresponding errors of the original individual mappings, independent from the order in which the mappings were combined. The maximum and particularly the minimum tag mapping errors were even reduced in the process, which we consider evidence for the feasibility of our approach. We conclude that the collaborative mapping prototype provides an encouraging example for the general idea of employing super-distributed smart entities as a substrate for the realization of collaborative activities.

As part of future work, means for performing the dead reckoning itself with the help of a purely SDRI-based middleware service should be investigated in order to free the MoD from its dependence on the rotation sensors. To date we developed and simulated several dead reckoning techniques based on single- and multi-reader configurations (see [Zwe04] for preliminary results). In addition, the mapping system should be further developed to make use of the *Location* abstrac-

tion provided by our Location Manager implementation in order to improve the robustness against individual tag failures.

Acknowledgments

We wish to thank Vito Piraino for his work on the implementation of the SDRI middleware prototype [Pir04]. We also wish to acknowledge Nicola Oprecht for his work on the implementation of the SDRI Tracking and Positioning prototype [Opr05], and Marco Bär for his work on the Collaborative SDRI Mapping system [Bär04]. We further acknowledge Thomas Zweifel for his help in the implementation and simulation of RFID-based dead reckoning techniques [Zwe04].

14. iPOS: Fault-Tolerant Self-Positioning with QoS Guarantees Based on Multi-Sensor Data Fusion

In this chapter, we describe the architecture of iPOS (short for iPAQ POsitioning System), a fault-tolerant self-positioning system with quality-of-service guarantees for resource-limited mobile devices [Boh07a]. The architecture is based on a probabilistic data fusion algorithm that is capable of efficiently combining the location information obtained from an arbitrary number of location sensors. As proof of concept, we present a prototypical implementation and discuss an experimental evaluation of the iPOS system.

14.1. Motivation and Background

Ubiquitous computing technology has become increasingly wide-spread in public places and even in private homes. For instance, wireless communication infrastructures such as Wireless LAN (WLAN) or Bluetooth have become quite popular in airports, train stations, and public buildings, and not only as substitutes for expensive wired networking [GBPK02]. Simultaneously, other typical pervasive computing technologies such as radio frequency identification (RFID) have become commonplace in industrial and commercial facilities, such as in manufacturing plants, museums and department stores [RFI03].

14.1.1. Location-Aware Computing and Positioning

Considering emerging ubiquitous computing applications, context awareness has become a major field of research [CK00]. Here, the general idea is that a broad range of applications may benefit from the capability of adjusting to their current situation and to the prevailing circumstances [Nel98, DAS01]. Context awareness is particularly important to mobile devices, whose local computing context, physical context, and user context is subject to change frequently and spontaneously (see Chapter 9.1). Furthermore, location information has been identified to remain "the single most important piece of context used in ubicomp applications" [ABO02]. This is mirrored in the large number of mobile location-aware ubiquitous computing systems [HB01] that have been conceived over the years, and in the ongoing research efforts in the area [SLP05]. Many of these systems require knowledge about the position of mobile objects and devices, as it is the case with the Lancaster mobile tourist guide [DCME01] and other indoor navigation systems [Son98, RS00].

While outdoor positioning systems are mainly based on cellular networks, satellite-based technologies [KOB01], and radio signal propagation, many indoor location systems rely on different technologies, often of a single kind, such as on Wireless LAN signal strength measurements [BP00], radio beacons [KWS02], video analysis [RS00], infrared beacons [Son98], or ultrasonic sound [WJH97].

14.1.2. Dependability Challenges

There are several issues that have to be addressed by location-aware systems.

Firstly, computing systems for location awareness and positioning that are solely based on one technology are prone to service disruption and interferences, because the unavailability or failure of the single underlying technology leads to a complete failure of the service as a whole. Therefore, the dependence on a single type of hardware can be considered a major drawback with respect to reliability and availability aspects.

Secondly, many currently available location systems do not fully exploit the sources of location information that are readily available in existing ubiquitous computing environments. Thus these location systems are not able to take advantage of various new sources of location information that are – as a side-effect – implicitly provided by the infrastructure. While technologies such as RFID, WLAN or Bluetooth may have been invented with a certain primary application in mind (e.g., to be used for object identification or wireless data transfer), these technologies often leak location information during operation. Furthermore, the majority of existing location systems require the deployment of specialized hardware, such as grids of customized infrared or ultrasonic beacons, GPS receivers, etc., leading to additional costs.

Thirdly, the number of small microcomputer-equipped objects and mobile devices is about to increase significantly with the realization of the ubiquitous computing vision. If these devices have to be enabled to benefit from new positioning and location sensing techniques, it is particularly important to cope with the resource limitations and constraints these devices impose on potential positioning systems. Therefore, there is a strong need for adaptive and efficient positioning mechanisms that take into account the shortcomings of small, resource-limited devices.

14.1.3. Contribution

We present a robust and scalable probabilistic positioning system which addresses a number of shortcomings of common ubiquitous computing location systems. The positioning system is based on a novel sensor modeling technique in combination with a lightweight multi-sensor data fusion architecture, which is explicitly tailored to operate efficiently and autonomously as a stand-alone service on small resource-limited devices. The fusion architecture is modularized and supports the integration of an arbitrary number of sensors and external third-party positioning services. Further, our positioning system is suited to work with standard off-the-shelf sensor hardware that is typically found in ubiquitous computing environments, thus allowing to exploit existing ubiquitous computing infrastructures for positioning. A special feature of our developed data fusion algorithm is that it provides, under certain conditions, applications with quality-of-service (QoS) guarantees.

14.2. Fundamentals

There are many systems available that use dedicated hardware infrastructures for localization and positioning. The most widely used system today is the Global Positioning System (GPS), which is restricted to outdoor positioning. Since the use of dedicated hardware is often very costly, both for the infrastructure and dedicated components in end-user devices, it is desirable to use features of already existing installations for localization. Communication technology such as wireless networks (WLAN, Bluetooth) or identification systems (RFID, barcodes) come in handy for that purpose. One of our goals is the exploitation of this existing infrastructure for determining the location of objects. Features of wireless communication such as signal propagation time or signal strength are promising candidates. Identification systems where the location of one component is known (such as a stationary RFID reader) provide us with immediate position information.

In our system, we distinguish two types of location information according to Hightower and Boriello [HB01]: (1) *physical positions* (or *geographic positions*) that are represented by a tuple of spatial position coordinates with regard to a well-defined local or global coordinate system, and (2) *symbolic locations*, which describe abstract ideas of where something is, such as "in the office" or "next to the printer".

14.2.1. High-Level Sensor Fusion

Basically, there are three possible levels on which to perform sensor fusion [HL01]: on raw sensor data, on features extracted from raw data, and on the decision level.

Fusion on *raw sensor data* is only possible if the domain of all sensors is the same, i.e. they are of the same type and measure the same quantity. In our approach, the fusion of raw sensor data is supported, but the fusion process has to be performed within a module representing a sensor, and the outcome of that process would be regarded as the measurement of a single (logical) sensor. As an example, consider multiple RFID tags located on the same object. Each tag, when recognized by an RFID reader, would provide a location of the object. But since this location is the same for all tags—the location of the RFID reader—the locations can simply be collapsed into one.

Feature extraction is a technique that reduces the amount of data produced by a sensor and abstracts away all information that is irrelevant for the task at hand—in case of a positioning system, only information relevant to determining the current location is retained. Multiple sensors (working on different domains) can be combined after relevant features have been extracted from the raw data. On the one hand, feature extraction for RFID tags makes no sense, since RFID tags immediately yield an object's location. Therefore, data obtained from RFID tags and WLAN data cannot be fused on the feature level. On the other hand, for data obtained from Bluetooth and WLAN receivers such as signal strength information, feature level fusion may be suitable. In its current state, our system does not support data fusion on the feature level directly, since no component for feature extraction and combination is incorporated. However, data fusion on the feature level can again be accomplished in the sensor modules themselves.

In our system, sensor fusion is performed mainly on the decision level, in so

far that each sensor module provides the system with a set of possible values (object locations) represented as a probability distribution. These distributions are combined to compute a new probability distribution that represents the most likely location of the object. This approach facilitates a modular and extensible system architecture. The number and different classes of sensors are not limited. Processing the sensor data can be performed remotely (i.e., not on the object itself) and pushed to the object in the form of an internal location event. When single sensors fail (or are shut off due to power saving efforts), the quality of the localization is affected, but the system as a whole remains functional. If the quality of the positioning dropped to room-level accuracy, for instance, it would still serve applications for which it is sufficient to get regular updates on the position whenever the user enters another room, being indifferent to all intermediary positions of the user.

14.2.2. Map Knowledge

The existence of a map model is an important aspect in our approach. The map model serves two main purposes. First, it provides a frame of reference within which all sensor data is interpreted and combined with help of a data fusion algorithm. Second, it provides applications with symbolic and sub-symbolic location information, such as "near table" or "at coordinates (x, y) in room Z", respectively.

In our system, a map is internally represented as a 2-dimensional *grid* with a fixed cell size where the cell size can be chosen individually for each map. A cell is represented by a data structure, which contains the following information:

- a value for the probability that the mobile device is located within this cell (cell occupancy probability);
- probability values for movement into the eight adjacent cells, and for staying in the cell (transition probabilities).

There are also cells of a special type (inherited from the standard type) that mark the transition to another map; this allows switching to other maps when the object leaves the area of the current map.

In the domain of mobile robotics, the use of probabilistic occupancy grids for positioning and navigation is an established technology, which has been successfully employed [Elf89, Bal96, BFH97]. A particular strength of the grid-based approach is that it allows us to model the uncertainty of location information, which is induced by *mobility* of people and their devices, in an intuitive way. While for stationary objects a calculated position retains its validity over time, this is not the case for moving devices. Instead, the significance of a calculated position diminishes more rapidly the faster the device is moving, and the uncertainty of what the true position actually is increases accordingly if no further location information is available. Further, the use of a grid facilitated the development of a fusion algorithm that is sensitive to the topology of different places: the algorithm respects walls and obstacles by setting the corresponding cell transition probabilities to zero in the grid.

In our system, we model the growing uncertainty of previously calculated positions in the absence of further sensory input by repeatedly recalculating cell occupancy probabilities on our cell grid according to a simple motion model.

The map model is used throughout the localization process. In a learning or initialization phase, symbolic locations are linked to map coordinates. During usage of the positioning system, map knowledge is exploited in various ways. For instance, walls are considered during the recalculation of cell occupancy probabilities. When a new position is to be determined, the outcome of the positioning process generally is a list of locations, ordered by probability. Again, map knowledge can be exploited in order to eliminate certain points from that list: a point can be located outside of a building or blocked by walls and therefore be unreachable (plausibility validation), or exceed a maximum allowed distance from the last valid position (outlier detection).

14.3. System Architecture

In the following, we describe the design goals and architecture of our map-based, multi-sensor data fusion system that enables a fault-tolerant self-positioning of mobile, resource-limited devices.

14.3.1. Design Goals

The mobile positioning system we developed and prototypically implemented realizes the following design goals:

- *Fault Tolerance:* The system is capable of tolerating the temporary or permanent failure of individual location sensing components (*location sensors*). This is achieved by means of a redundant fusion architecture that enables the system to exploit the redundant heterogeneous sources of location information found in the vicinity of the mobile device performing the self-positioning.
- *Adaptability:* The fusion architecture is designed to perform an adaptive resource management. It enables the system to dynamically load or unload location sensing components during runtime, according to availability and coverage.
- *Self-Sufficiency:* The positioning system is in a position to operate in a self-sufficient manner on a mobile device (MoD) without the need of a background service infrastructure or a centralized remote server by using locally available resources.
- *Extensibility:* The modular design of our sensor fusion architecture makes it possible to easily integrate additional location sensors and location technologies at a later point in time.
- *Interoperability:* By using a uniform internal representation of location information our systems enables the integration of arbitrary third-party positioning services.
- *Versatility:* The positioning system is capable of integrating both geographic position information and symbolic location information. Symbolic location information is resolved into geographic information with the help of map models. Local and global geographic position information can be processed

by means of coordinate system transformations and the use of the global WGS-84 [Eur06] standard as reference system.

- *Quality-of-Service Guarantees:* By discriminating "reliable" from "unreliable" location sensors in our model, our fusion algorithm is in a position to provide certain quality-of-service guarantees during operation.
- *Support for Resource-Limited Mobile Devices:* We provide a lightweight implementation of our fusion architecture that proved to perform well on small, resource-limited MoDs.

14.3.2. Architecture Overview

The positioning system we developed is executed on the MoD whose position is to be established during operation. The main architectural components of the positioning system are the resource manager, the fusion engine, the map handler, the internal event abstraction layer, and the sensor plugins (see Fig 14.1).

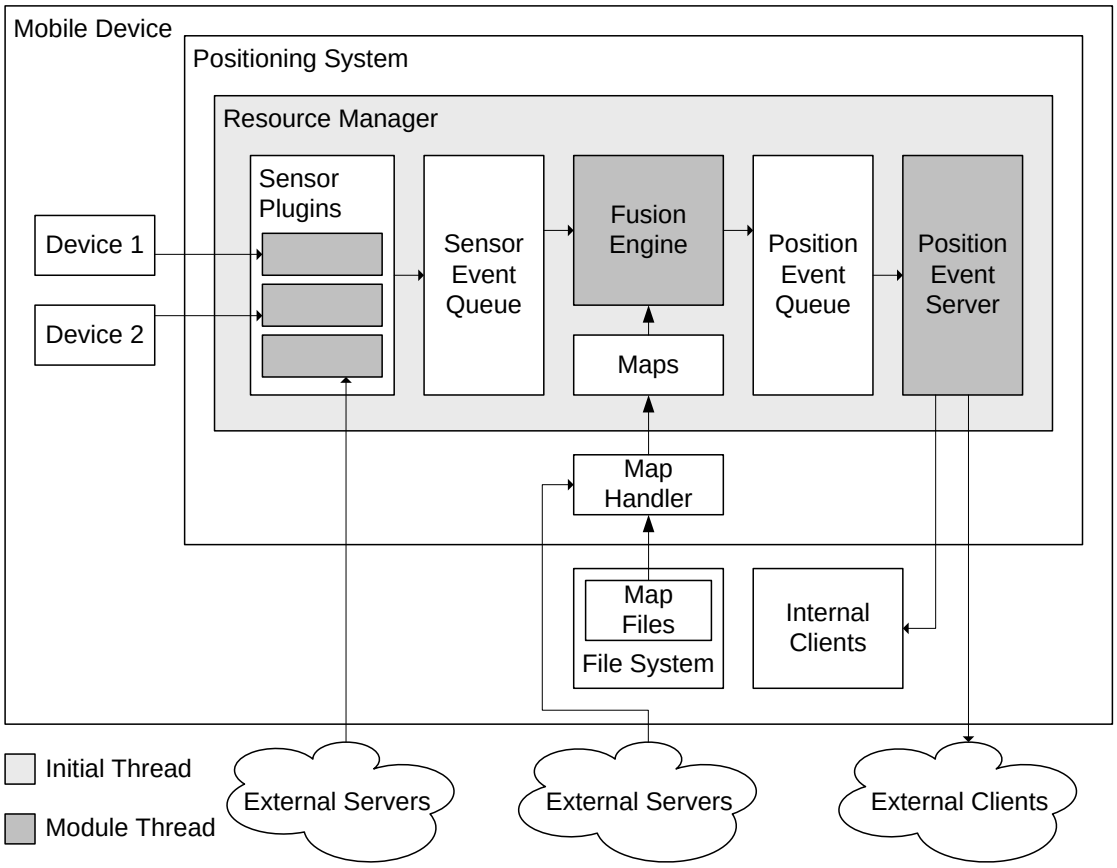

Figure 14.1.: Overview of the architecture of the positioning system

In the context of our work, a *sensor* may be either a physical device or an application that in turn preprocesses the location information of other physical devices or applications. The sensor hardware or application can either be part of the MoD itself or constitute an external resource, such as a remote server, for instance. A *sensor plugin* is associated with one specific type of sensor. It preprocesses and transforms sensory location information into an abstract representation of position

estimates we call *sensor events*. Each sensor event contains an absolute position with respect to a given two-dimensional map model. New sensor events generated by sensor plugins are written into an internal *sensor event queue* for later retrieval and further processing.

The core component of the system is the *fusion engine*, which processes sensor events to calculate the current position of the MoD. During each iteration of the positioning calculation, the fusion engine takes out the most recent sensor events from the sensor event queue and combines them by means of a map-assisted probabilistic sensor-fusion algorithm. The maps required in the process are obtained with the help of the *map handler* component, which can retrieve maps stored on the local file system or download them from a remote server. The output of the fusion engine are time-stamped *position events*, which are stored in the position event queue. From that queue, the *position event server* takes out the most recent position estimates and makes them available to local or remote clients by means of a socket-based querying interface.

The *resource manager* is responsible for managing the different components of the positioning service, including the initialization, loading and unloading of components during runtime. In particular, the resource manager controls the operation of the sensor plugins, initializes the event queues and the map handler, starts the position event server, and controls the operation of the fusion engine.

14.3.3. Map Model

The map model we used in our system consists of a two-dimensional equidistant cell grid with square cells of equal size (e.g., cells of 0.5 m^2 or 1.0 m^2). Each cell within a grid is defined by an unambiguous cell index, and by a set of geographic position coordinates. The position coordinates may be local with regard to the map, or global according to the WGS-84 standard [Eur06]. In our current implementation, it is assumed that the map is planar with a fixed z-coordinate for all cell position coordinates in a particular map. The specification of two global coordinates for two local reference positions in a map defines a coordinate transformation between local and global positions.

Each cell further contains a set of probability values for the transition to up to eight neighboring cells, where each value can be set to zero for indicating obstacles or walls. A special type of cells called *transition cells* are used to connect a map to other maps. The map model further supports the placement of *objects* onto cells. An object has a unique symbolic identifier and a geographic position which is determined by the cell onto which the object is placed. It is possible to create arbitrary types of objects, each of which contains a number of specific attributes.

For instance, an object of type "Radio Beacon" may represent a physical radio transmitter and contain attributes for its ID and its transmission range in meters, whereas an object of type "WLAN-RSS-Tuple" may represent a symbolic location identifier that is associated with a received signal strength (RSS) measurement of nearby Wireless LAN access points that was performed at the corresponding physical position, together with attributes describing the configuration of the measuring tool.

The map model also provides a set of basic methods for processing maps, such as for joining and intersecting areas of a map, and for looking up and modifying

cells or objects within a map. For facilitating map operations, the map model introduces a *Region* abstraction, which is defined as a set of cells, and the *Clip* abstraction, which is defined as the *Region* containing all cells that have an occupancy probability greater than zero concerning the current location of the MoD. The main idea of the *Clip* is to narrow down the number of significant cells that have to be considered during the map-assisted fusion process in order to enable an efficient position calculation on resource-limited mobile devices.

14.3.4. Sensor Event Model

Sensor plugins generate *sensor events*. We say that a sensor event *fired* when it was created by a sensor plugin.

For each type of sensor, a separate class of sensor events is defined that share common global properties. A sensor event contains location information in the form of either a symbolic location identifier or a geographic position. A sensor event containing symbolic location information we refer to as *symbolic sensor event*. Likewise, a sensor event containing geographic position information we call a *geographic sensor event*.

The location information provided with a sensor event defines the (geographic) *center point* of that sensor event. For sensor events featuring a symbolic location, the corresponding center point is determined with help of the map model described earlier in Sect. 14.3.3. For that, each class of symbolic sensor events is assigned a separate class of objects in the map model. Then each symbolic sensor event of that class is linked to an object of the corresponding object class with the same individual symbolic identifier. Any such object has to be added to the respective grid map during the learning or initialization phase of the positioning system. Obviously, the mapping of sensor events to objects in the map model is bijective. Therefore, with the help of the map model, it is possible to decide a priori which symbolic sensor events can possibly occur within the mapped area. In particular, it is also possible to determine the symbolic sensor events that did not fire (i.e., where not created by sensor plugins) during an iteration of the sensor event fusion process. We exploit this ability to increase the accuracy of our position calculation (see Sect. 14.4).

Concerning shared properties, each class of sensor events contains a *range* property. Together with the individual center point, the range defines the spatial area of influence of a single sensor event in the two-dimensional map model, which corresponds to a *Region* in the map model. Thus the range of a sensor event class is a measure of the *accuracy* of the corresponding sensor plugin and its provided location information. In the following, whenever we speak of the intersection or union of sensor events, we refer to the intersection or union of the respective *Regions* spanned by these sensor events. We also say that a sensor event *covers* an individual cell if that cell is part of the *Region* of the sensor event. Each sensor event class further contains a property *blocking*, which indicates whether the area of influence of the sensor event is affected by obstacles or walls.

For instance, a sensor plugin for sensing nearby radio beacons may create a sensor event containing the symbolic identifier of a detected beacon as symbolic location identifier. Then, by means of the map model in which a corresponding beacon object was placed earlier and named accordingly, the geographic position of the

cell containing the beacon object is taken as the geographic position of the radio beacon. Alternatively, in case the detected beacon already transmits its physical position, no further map lookup is required to establish the center point of the corresponding sensor event. For example, given a maximum range of 10 m for the type of radio beacon, the area of influence of the sensor event during the map-based fusion process is determined by identifying all cells within that range, starting from the center point. In the process, if the sensor event is blocking, only cells that are in line of sight from the center point are considered.

14.3.5. Las-Vegas and Monte-Carlo Sensor Events

Depending on the type of sensor, position estimates vary in terms of accuracy and reliability. In our system, we discern two categories of sensor plugins: (1) *unreliable* sensor plugins, which we call *Monte-Carlo* sensor plugins, and (2) *reliable* sensor plugins, which we call *Las-Vegas* sensor plugins. Likewise, we refer to a sensor event generated by a Monte-Carlo or Las-Vegas plugin as Monte-Carlo or Las-Vegas sensor event, respectively.

The semantics of the Las-Vegas and Monte-Carlo sensor plugins follow the semantics of the behavior of randomized algorithms. A *Monte-Carlo sensor plugin* (in short: *MC-plugin*) shows a *deterministic* behavior in so far that it always returns a result upon request, in our case location information encapsulated in a Monte-Carlo sensor event (in short: *MC-event*), but the resulting position information is liable to be erroneous and *possibly false*. In contrast, a *Las-Vegas sensor plugin* (in short: *LV-plugin*) displays the following *indeterministic* behavior: it does not always return a Las-Vegas sensor event (in short: *LV-event*), but if it does, the provided location information is *correct* in the following sense: it is guaranteed that the actual current position of the MoD lies within the boundaries determined by the known accuracy and area of influence of the respective Las Vegas sensor event.

A typical class of LV-plugins are plugins that detect the presence of radio beacons for which the maximum range (i.e., the maximum distance from which a beacon can still be detected) can be safely and accurately determined. Consequently, if a beacon of that type is detected, the current position of the MoD is known to be within the area of influence of the beacon (i.e., within the physical area in which the beacon can be detected). The knowledge of the maximum range yields an upper bound for the positioning error (which equals the maximum distance between any two physical positions within the range of the beacon). The upper bound for the positioning error can be interpreted as an upper bound for the *accuracy* value of the sensor (a higher metric accuracy value actually means a lower accuracy).

Based on this model, if multiple beacons (which may belong to different classes and therefore feature different ranges) are detected by the corresponding LV-plugins, then the current position of the receiver (i.e., the MoD) by definition has to lie in the intersection of the individual areas of influence of the involved beacons. In the map model, this results in a *Region* that is equal to or smaller than the *Region* covered by the beacon with the smallest range. This implies that the resulting accuracy for the intersection of multiple beacons is higher (i.e., better) or equal to the one of the beacon with the highest accuracy (and with the lowest accuracy value). A further implication of the reception of multiple LV-events is

that the current *Clip* (i.e., the *Region* containing all cells that are candidates for the current position of the MoD, see Sect. 14.3.3) can be narrowed down to the intersection of the *Regions* of all received LV-events.

A typical example for MC-plugins are plugins that rely on received signal strength (RSS) measurements of all detected senders in the vicinity of the MoD, and which determine the best match out of previously empirically learned RSS patterns for a number of reference positions by means of a nearest neighbor metric, for instance. Obviously, in case the signal strength values for some of the senders vary significantly (e.g., due to interference or mobile obstacles temporarily blocking or reflecting the signals), the calculated best match possibly yields a reference position that is far off the true best match for the actual current position of the mobile device. In the absence of LV-events, due to the inherent inaccuracy of MC-events, the *Regions* of any detected MC-events have to be joined with the current *Clip* to obtain the new *Clip*. In the best case, the new *Clip* equals the old one in case no new cells were added. In the worst case, if the *Regions* of the MC-events and the *Clip* are pairwise disjoint, the Clip grows by the number of cells contained in the *Regions* of the corresponding MC-events.

14.3.6. Probability Distributions for Las-Vegas and Monte-Carlo Sensor Events

For each type of sensor plugin and associated sensor event class, a probability distribution has to be defined that describes the likelihood that the MoD is located at a certain cell in the grid map model given that a certain sensor event was detected. We modeled this probability distribution in the form of $p(e, c)$, where c represents the cell for which the position probability has to be calculated, and e the sensor event that was detected. The probability function for each sensor plugin (or the associated sensor event class) can be derived from a formal model (e.g., based on signal propagation), or from empirical measurements (e.g., based on signal strength measurements at different physical locations).

Once the probability distribution functions for the sensor event classes have been defined, the probability that the MoD is located at a cell c if a *reliable LV-event* e^{LV} was observed is given by

$$P(\text{“MoD is located at cell } c\text{”}|e^{LV}) = \begin{cases} p^{LV}(e^{LV}, c), & \text{if } covers(e^{LV}, c) = true; \\ 0, & \text{otherwise.} \end{cases}$$

$covers(e^{LV}, c)$ is a predicate that evaluates to *true* if cell c is an element of the *Region* of cells spanned by the given LV-event e^{LV}, and *false* otherwise. We can see that whenever the predicate is false because cell c is outside the area of influence of a detected reliable sensor event, then the position probability for that cell c and the given LV-event e^{LV} is zero.

The probability that the MoD is located at a cell c given an observed *unreliable MC-event* e^{MC} is given as

$$P(\text{“MoD is located at cell } c\text{”}|e^{MC}) = p^{MC}(e^{MC}, c) \geq \delta, \text{with } \delta > 0.$$

In the case of unreliable MC-events, the position probability for any cell c derived from an unreliable sensor event is always greater than zero. This reflects the un-

certainty about the error-prone location information provided by MC-events. Furthermore, keeping MC-probabilities greater than zero ensures that position probabilities obtained from unreliable MC-events can be safely multiplied with position probabilities obtained from reliable LV-events without nullifying the product (see fusion engine below). In our system, the value of δ is defined as a lower bound for probability values (minimum probability unequal to zero) – all probability values in the grid model that fall below the value of δ are treated as probability zero.

14.4. Probabilistic Sensor-Fusion Algorithm

In this section, we describe the formal mathematical models and elements that constitute the foundation of our sensor-fusion algorithm used for positioning.

14.4.1. Calculation of the *Clip*

To recalculate the *Clip* between one iteration of the positioning procedure and the next, first any available LV-events are considered, and then the obtained MC-events.

1. *Processing of Las-Vegas Events:* According to the modeling of LV-events, the MoD is known to be located within their spheres of influence. Therefore, if LV-events are available during an iteration of the positioning system, the *Clip* containing the cells that are candidates for the current position of the MoD is recalculated as the *intersection* of the *Regions* spanned by the individual LV-events. Afterwards, the occupancy probabilities in the new clip are normalized to sum up to 1.

2. *Processing of Monte-Carlo Events:* Since MC-events are inherently unreliable, they cannot be used to narrow down the *Clip* during the computation of the position of the MoD. Instead, depending on the availability of LV-events during a single iteration of the event fusion process, the Clip is updated according to the following rules:

   a) *No LV-events available:* The *Clip* is recalculated as the *union* of all cells contained in the *Regions* of the available MC-events and normalized.

   b) *Some LV-events available:* The *Clip* obtained in step 1 is used as the new *Clip* without further modification.

14.4.2. Processing Las-Vegas Sensor-Events

For each cell c in the *Clip*, the probability that the MoD is situated in cell c at iteration t (denoted as $C_t = c$) is calculated using the available LV-events. We call this probability *LV-probability*, and denote it with term $P^{LV}(C_t = c)$.

Let e_i^{LV}, $i = 1, \ldots, k$, be k LV-events that fired and cover a cell c in the map model, and let e_i^{LV}, $i = k+1, \ldots, n$, be $n - k$ symbolic LV-events that did *not* fire but which also cover cell c. Further, let $p_i^{LV}(e_i^{LV}, c)$ be the position probability of cell c given that LV-event e_i^{LV} has fired. Then we calculate the LV-probability $P^{LV}(C_t = c)$ as shown in Eqn. 14.1.

$$P^{LV}(C_t = c) := P_F^{LV}(C_t = c) * P_N^{LV}(C_t = c), \tag{14.1}$$

with

$$P_F^{LV}(C_t = c) := \prod_{i=1}^{k} p_i^{LV}(e_i^{LV}, c) \tag{14.2}$$

and

$$P_N^{LV}(C_t = c) := \prod_{i=k+1}^{n} \left[P(fires(e_i^{LV}, c) = false)\right]. \tag{14.3}$$

In the process, Eqn. 14.2 fuses all fired LV-events that cover cell c by calculating the product of the respective probabilities. The result is multiplied with the term displayed in Eqn. 14.3. The latter term represents a *negative feedback*, penalizing the known existence of symbolic LV-events covering cell c that did not fire, thus reducing the overall position probability of cell c. Ideally, the probability for an LV-event e_i^{LV} to fire for a cell c that is covered by the sensor event when the MoD is located at that cell should equal one (Eqn. 14.4).

$$P(fires(e_i^{LV}, c) = true) = 1, \text{with } covers(e_i^{LV}, c) = true. \tag{14.4}$$

However, in practice, the location sensing performed by the sensor plugins is liable to be negatively affected by technical or environmental disturbances, such as signal interference or signal blocking, for instance. To account for such disturbances, we introduced an uncertainty factor ϵ_i^{LV}, which models the probability that a sensor event LV_i that should fire under ideal conditions given that the MoD is located at cell c does *not* fire:

$$P(fires(e_i^{LV}, c) = true) := 1 - \epsilon_i^{LV}, \text{with } covers(e_i^{LV}, c) = true, \tag{14.5}$$

$$\text{which can also be phrased as } P(fires(e_i^{LV}, c) = false) = \epsilon_i^{LV}. \tag{14.6}$$

The value of ϵ_i^{LV} has to be established for each sensor plugin, either with the help of a theoretical model or by means of empirical measurements (calibration). Generally, the probability that a sensor event does not fire even though the MoD is located on a cell in range of that sensor event should be comparably small to ensure the usefulness of the corresponding plugin: $0 < \epsilon_i^{LV} \ll 0.5$.

Table 14.1 provides a summary of the probabilities that an LV-event e^{LV} fires according to its area of influence, assuming that the MoD is located at cell c of the grid map model. The table also summarizes how this knowledge is exploited in our system.

$covers(e^{LV}, c)$	$P(fires(e^{LV}, c) = true)$	$P(fires(e^{LV}, c) = false)$	Purpose/Benefit
$true$	$1 - \epsilon^{LV}$	ϵ^{LV}	Fault-tolerant sensing
$false$	0	1	*Clip* recalculation

Table 14.1.: Probability that the LV-plugin of type LV running on the MoD located at position c creates the Las-Vegas sensor event e^{LV}

14.4.3. Processing Monte-Carlo Sensor-Events

For each cell c in the *Clip*, the probability $P^{MC}(C_t = c)$ that the MoD is situated in cell c at iteration t (denoted as $C_t = c$) is calculated based on the available MC-events. We call this probability *MC-probability*.

Let e_i^{MC}, $i = 1, \ldots, m$, be m MC-events that fired and cover a cell c in the map model. Further, let $p_i^{MC}(e_i^{LV}, c)$ be the position probability of cell c given that MC-event e_i^{MC} has fired. Then the MC-probability $P^{MC}(C_t = c)$ is calculated according to Eqn. 14.7.

$$P^{MC}(C_t = c) := P_F^{MC}(C_t = c), \tag{14.7}$$

with

$$P_F^{MC}(C_t = c) := \prod_{i=1}^{m} p_i^{MC}(e_i^{MC}, c). \tag{14.8}$$

The unreliability of MC-events implicates that there is no reliable information on potential MC-events that also cover cell c but did *not* fire. Consequently, we do not find a term $P_N^{MC}(C_t = c)$ in Eqn. 14.7.

14.4.4. Probabilistic Map-Based Fusion of Position Information

The LV-probabilities and MC-probabilities calculated from LV- and MC-events during iteration t are fused with the existing position probabilities $P(C_{t-1} = c)$ that were calculated during the previous iteration with index $t - 1$. This is achieved by multiplying the probabilities $P^{LV}(C_t = c)$ and $P^{MC}(C_t = c)$ obtained from Eqn. 14.1 and 14.7, respectively, with the old position probability $P(C_{t-1} = c)$ for each cell c in the *Clip*:

$$P(C_t = c) := P(C_{t-1} = c) * P^{LV}(C_t = c) * P^{MC}(C_t = c) \tag{14.9}$$

The position probabilities $P(C_{t-1} = c)$ of previous iterations are stored in the respective cells of the map model. These probabilities are generally referred to as *cell occupancy probabilities*. The cell occupancy probability of a cell c in the grid model states the likelihood that the MoD, which is to be positioned, is located at that cell.

By fusing the occupancy probabilities of previous iterations with the newly obtained location information, a continuity of the positioning is achieved, assuming that subsequent positions of the MoD are spatially proximate. Further, it enables the system to provide position estimates even in case no sensor events are available.

An invariant of the grid map model is that the overall occupancy probability values for all cells sum up to one. In our case, that means that all the occupancy probabilities of the cells in the *Clip* sum up to one, as the *Clip* by design contains all the cells of the active maps that feature a non-zero occupancy probability.

For that reason, at the end of each iteration of the probabilistic map-assisted data fusion procedure, any new obtained cell occupancy probabilities are stored in the respective cells of the corresponding map models and normalized to sum up to one. The normalization effort is limited to all cells contained in the *Clip*, as the occupancy probabilities for all other cells outside the *Clip* are zero (which

is taken care of on the fly by the methods provided for managing the *Clip*). The methods operating on the *Clip* further guarantee that all cells in the *Clip* maintain a probability greater or equal to a threshold $\delta > 0$ when new cells are added to the clip or when probability values decrease during the fusion process. This maintenance may also require that the threshold $\delta > 0$ is dynamically adapted to the lowest probability value of a cell found in the *Clip*.

14.4.5. Mobility Heuristic in Absence of Sensor Events

In case that there is no new location information in form of sensor events available during an iteration step, we update the cell occupancy probabilities in the map model according to a simple mobility model and a mobility heuristic.

The mobility heuristic works as follows: With the help of the cell transition probabilities stored in each cell, we repeatedly recalculate the cell occupancy probabilities (see Sect. 14.2.2) for each cell in the *Clip*. Let C_{t-1} be the position of

$n_{2,j}$	$n_{3,j}$	$n_{4,j}$
$n_{1,j}$	x_j $= n_{0,j}$	$n_{5,j}$
$n_{8,j}$	$n_{7,j}$	$n_{6,j}$

Figure 14.2.: Cell $x_j = n(0, j)$ and its neighboring cells $n_(i, j)$, $(1 \leq i \leq 8)$

the MoD at iteration $t-1$ and $n_{i,j}$ $(1 \leq i \leq 8)$ the neighboring cells of cell x_j (see Figure 14.2). Then the mobility heuristic function is applied to calculate the new cell occupancy probability $P(C_t = x_j)$ of each cell x_j in the current *Clip* as described in Equation 14.10.

$$P(C_t = x_j) := \sum_{i=0}^{8} (P(C_{t-1} = n_{i,j}) * P(C_t = x_j | C_{t-1} = n_{i,j})) \tag{14.10}$$

In the equation, $P(C_t = x_j | C_{t-1} = n_{i,j})$ is the probability of a transition from the neighboring cell $n_{i,j}$ at iteration $t-1$ to cell x_j at iteration t, while $P(C_{t-1} = n_{i,j})$ denotes the cell occupancy probability of cell $n_(i, j)$ at iteration $t-1$.

Initially, the transition probabilities for each cell are defined as follows: let w be the number of neighboring cells (including the cell itself) that can be reached, then the transition probability for these transitions is uniformly set to $\frac{1}{w}$, favoring no particular direction. For all remaining directions, e.g., in case the neighboring cell is blocked by a wall or does not exist, the transition probability is set to zero. Later, these cell transition probabilities can be adjusted according to characteristic personal movement patterns acquired during a learning phase, for example, as we investigated and prototypically implemented in [Sch03].

The number of executions r of the above mobility heuristic on the cells of the *Clip* is determined separately for each map that is involved, based on the specified mobility model. The *mobility model* comprises parameters for the assumed average movement speed v of the MoD, and for the currently set frequency of position calculations f (e.g., 1 Hz). Based on these parameters, the metric distance d is calculated as $d := \frac{v}{f}$. The value d serves as an estimate for the distance the MoD may have moved while no location information was provided by the sensor plugins of the positioning system since the last iteration step. Then, given a map model with the side length a of its cells in the cell grid, the number of executions of the mobility heuristic on the respective map is calculated as $r := \lceil \frac{d}{a} \rceil$.

During the execution of the mobility heuristic, after the completion of each iteration, the *Clip* is updated as follows: every cell outside the *Clip* that has been allotted a sum of probability values greater than the current minimal probability threshold δ is added to the *Clip*. As a final step, after all iterations were performed, the updated occupancy probability values of all cells in the *Clip* are normalized.

Literally speaking, the application of the mobility heuristic ensures that the cell occupancy probabilities in the grid wear off over time when no sensor events are provided by the sensor plugins. The aim of the heuristic is to emulate the expected increase in uncertainty about the position of the MoD due to the absence of new pieces of location information.

14.4.6. Position Derivation

The final step of each iteration of the probabilistic sensor fusion algorithm is the derivation of the estimate for the position of the MoD. For that, we apply a *maximum-likelihood heuristic* that searches for the cell with the highest position probability (i.e., the highest cell occupancy probability) in the current *Clip*. Taking the most recent position estimate (cell c_{init}) as a starting point in the grid map model, the heuristic searches for the cell c_{max} with the highest occupancy probability $P(C_t = c_{max})$. In doing so, the heuristic only considers cells that are within a maximum distance d from cell c_{init} and reachable in the map model (i.e., not blocked by walls).

The value of d is determined by means of the *mobility model* described in Sect. 14.4.5: $d := \frac{v}{f}$, with the assumed average movement speed v of the MoD and the frequency of position calculations f as parameters.

14.4.7. Operation of Probabilistic Fusion Engine

As we have seen, the *probabilistic fusion engine* plays a pivotal role in our positioning system. Running on top of a two-dimensional map model, the fusion engine during each iteration step takes the sensor events stored in the sensor event queues, fuses their location information based on the algorithm described in Sect. 14.4, and stores the resulting position events in the position event queue (see Fig. 14.3).

The flow of control of the probabilistic fusion engine and the integration of the different parts of the fusion algorithm is shown in Fig. 14.4.

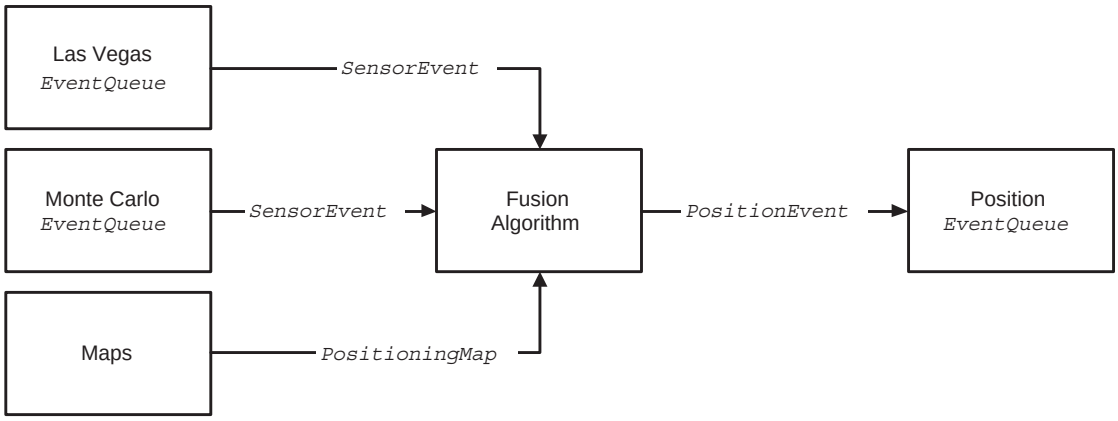

Figure 14.3.: Data flow of the probabilistic map-based position fusion procedure

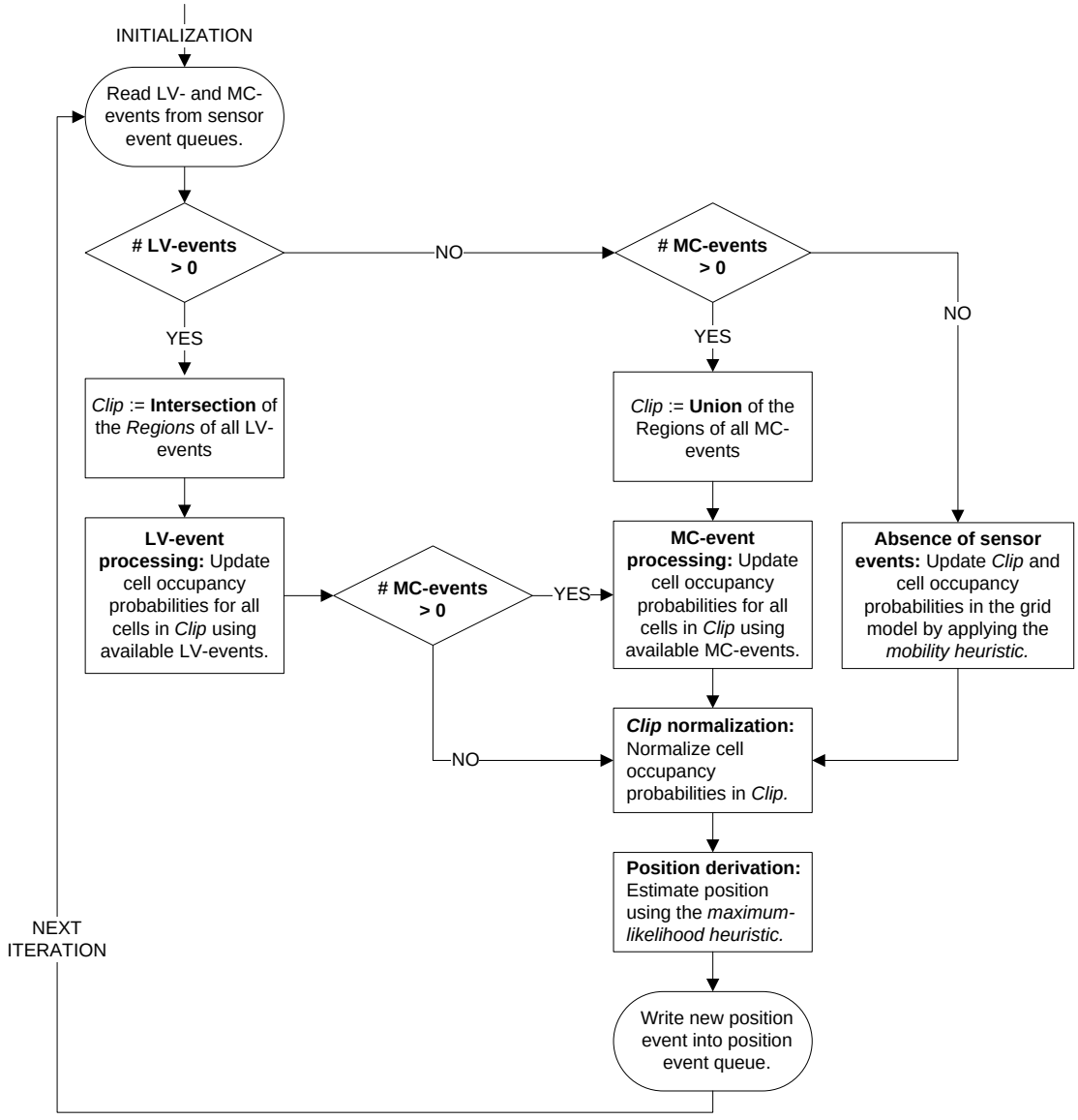

Figure 14.4.: Control loop of the probabilistic map-based fusion engine

14.5. Complexity Analysis

The complexity of one iteration of the fusion algorithm is determined by the number and type of available sensor events.

14.5.1. Fusion of Las-Vegas Sensor Events

Let $e_1^{LV}, \ldots, e_k^{LV}$ be k LV-sensor events obtained from LV-plugins $S_1^{LV}, \ldots, S_k^{LV}$ with ranges $n_1^{LV}, \ldots, n_k^{LV}$ (given in number of cells). Then the number of cells to be processed by the algorithm is defined by the size of the *Clip*, which by definition is obtained by intersecting the *Regions* of the LV-events. Consequently, the *Clip* is a subset of the smallest *Region* R_{min}, Clip $\subseteq R_{min}$, with $|R_{min}| \leq (2 * \min_{i=1}^{k}(n_i^{LV}))^2$. This means that in the case of LV-events, the number of cells to be processed by the algorithm is determined by the most accurate sensor event with a *Region* R_{min} containing at maximum $(2 * n_{min}^{LV})^2$ cells.

By determining the maximum possible range N^{LV} for an LV-event, $N^{LV} := \max_{i=1}^{k}(n_i^{LV})$, we can specify the upper bound $(2N^{LV})^2$ for the amount of cells to be processed. Defining constant c^{LV} as $c^{LV} := (2N^{LV})^2$, then the worst case complexity of *fusing k Las-Vegas sensor events* is $O(c^{LV})$ and therefore constant with regard to the number of necessary cell occupancy updates. The complexity in terms of *number of multiplications per cell occupancy update* is $O(k)$ in the worst case, as the number of multiplications per cell equals the number of obtained LV-events that cover that cell.

14.5.2. Fusion of Monte-Carlo Sensor Events

Let $e_1^{MC}, \ldots, e_j^{MC}$ be j MC-sensor events obtained from MC-plugins $S_1^{MC}, \ldots, S_j^{MC}$ with ranges $n_1^{MC}, \ldots, n_j^{MC}$ in number of cells. Further, let N^{MC} be the maximum range of all MC-events, $N^{MC} := \max_{i=1}^{j}(n_i^{MC})$.

If $k > 0$ LV-events were also obtained during the current iteration of the fusion process, the *Clip* size is not affected by additional MC-events. Accordingly, the worst case complexity of *fusing k Las-Vegas sensor events and j Monte-Carlo sensor events* remains constant with $O(c)$ number of cell occupancy updates, with $c := (2N)^2$ and $N := \max(N^{LV}, N^{MC})$. The complexity in terms of *number of multiplications per cell occupancy update* is $O(k+j)$ in the worst case, as the number of multiplications per cell c equals the number of obtained LV-events and MC-events that cover that cell.

If no LV-events were obtained during an iteration of the fusion process, the *Clip* is recalculated as the union of all *Regions* of the MC-events. With the maximum range N^{MC} for MC-events, we can specify the upper bound $j * (2N^{MC})^2$ for the amount of cells to be processed, as in the worst case the *Regions* of the j MC-events are disjoint. Defining constant c^{MC} as $c := (2 * N^{MC})^2$, then the worst case complexity of *fusing j Monte-Carlo sensor events* is linear with $O(j * c) = O(j)$ number of cell occupancy updates. The complexity in terms of *number of multiplications per cell occupancy update* is $O(j)$ in the worst case, as the number of multiplications per cell equals the number of obtained MC-events that cover that cell. However, the number of multiplications per cell is reciprocally proportional to the number of cells that are covered by the j MC-events (i.e., the *Clip* size), since a smaller

overlap of the *Regions* of the available MC-events results in fewer multiplications per cell.

14.5.3. Upper Bound for the Range of Sensor Events

Theoretically, sensor events can have ranges of arbitrary size. However, in practice, the ranges of sensor events remain within certain boundaries. The reason for that is that the contribution of a single sensor event to the overall fusion process becomes less significant and meaningful the higher its range is, because the occupancy probabilities get increasingly spread and unfocused with increasing range: a higher range goes hand in hand with a quadratic increase of the corresponding maximum *Region* spanned in the map model. Therefore sensor event ranges beyond a certain point are disadvantageous and unjustifiable, and it is desirable to limit the maximum range of sensor events. This is straightforward for sensing hardware that *per se* requires a limited range due to technical limitations.

Otherwise, if a physical sensor technology would require sensor events with a range higher than a desired maximum range $N_{\max}$, it is preferable to build a *logic* sensor plugin instead which *preprocesses* the physical sensor data (e.g., by combining multiple measurements) and then creates a sensor event with a lower, more appropriate range. In our case, for instance, we created an MC-plugin based on received signal strength analysis for long-range Bluetooth beacons (BTnodes). As a result, the Bluetooth RSS MC-plugin reduced the range of the sensor events to approx. 5 m as opposed to the range of approx. 30 m that the BTnodes feature when used as individual radio beacons.

We conclude that it is feasible to assume an upper bound $N_{\max}$ for the range of sensor events, independent of the physical properties of the sensing technology.

14.5.4. Practical Considerations

In Sect. 14.5, we argued that it is feasible to assume an upper bound $N_{\max}$ for the range of sensor events, and that under this assumption the sensor-fusion process has linear complexity. In Table 14.2, we listed various sensor technologies that are typically found in today's ubiquitous computing infrastructures, and for which we implemented sensor plugins to be used with our positioning system. For each sensor class, we have denoted the characteristic range in meters, and the corresponding range in number of cells for different cell sizes. Looking at the data, we can see that all the sensor types are suited to support a sensor event range between fifty centimeters and 20 meters, which we still consider practicable for our system. Therefore, in our case, we could determine a concrete upper bound $N_{\max}$ for the range of sensor events of $N_{\max} = \lceil 20 \text{ m}/\textit{cell size} \rceil = 20$ for a cell size of 1×1 m.

14.6. iPOS Positioning System Prototype

We prototypically implemented a fully functional prototype of the positioning system according to the system architecture presented in Sect. 14.3. We named the prototype system *iPOS*, which stands for **i**PAQ **PO**sitioning **S**ystem. iPAQ refers to the type of mobile device (MoD) we used for prototyping: the MoD was represented by a personal digital assistant (PDA) of type HP iPAQ, H5450 Series,

Sensor class	Range [m]	$n_{2.0}$	$n_{1.0}$	$n_{0.5}$
Barcode tags	0.5	1	1	1
Passive RFID tags	0.5–1	1	1	1–2
Bluetooth RSS	5	2.5	5	10
Bluetooth beacons	10/20*/100**	5/10/50	10/20/100	20/40/200
Active RFID beacons	3/10/20*/50**	2/5/10/25	3/10/20/50	6/20/40/100

Table 14.2.: Typical sensor event ranges per sensor class and range. n_l denotes the corresponding sensor event range measured in number of square grid cells with a cell border size of l meters. The range of sensor events obtained by a Bluetooth-RSS-based plugin is defined by the performance of the used algorithm and influenced by the characteristics of the particular distribution of Bluetooth senders in the environment. In our case, given a regular grid of Bluetooth beacons with 2 m side length, we obtained a range of 5 m for our Bluetooth RSS MC-plugin. The range of our active RFID equipment could be adjusted by configuring the transmission power of the reader and the sensitivity of the tags. For our positioning system, we considered a sensor event range of 10 m and below as optimal, ranges up to 20 m as acceptable (*), and ranges beyond 20 m as inappropriate (**)

running the PocketPC 2002 (WinCE 3.0) operating system. The iPAQ PDA featured wireless LAN connectivity and a PCMCIA slot extension.

The implementation of the iPOS system is based in parts on an earlier implementation of a probabilistic positioning system [BV03], which was running on a laptop computer and not yet fully optimized for the execution on resource-limited mobile devices. While we could reuse large parts of the map model and of the plugin system, we revised the fusion algorithm and introduced the concept of reliable and unreliable sensor events. This significantly improved the effectiveness and performance of the fusion procedure in the iPOS system by reducing the number of cells that have to be examined as potential candidates for the position of the MoD. Further, we provided a new resource manager component and event processing layer, as well as an improved programming interface for processing maps, sensor events, and occupancy probabilities.

For the iPOS prototype system, we employed passive and active sensing technologies for providing location information to the MoD. Here *passive* means that the MoD can make use of the passive sensing technologies in a self-sufficient manner without the need of relying on infrastructure-based services – the MoD performs and controls the sensing autonomously. In contrast, we call a sensing technology *active* when it is managed by a third-party entity, which is beyond the control of the MoD – an active sensing infrastructure senses the MoD and provides any acquired location information by means of a well-defined network service interface (e.g., a Web service or RMI interface).

14.6.1. Passive Location Sensing Infrastructure

The *passive sensing infrastructure* consisted of (1) active RFID beacons (i-Q8 tags by Identec Solutions), (2) Bluetooth-enabled sensor nodes (BTnodes, `www.btnode.ethz.ch`), and (3) densely distributed passive RFID tags. For the latter we used the existing SDRI prototype described in Sect. 13.3.1. An overview of the passive sensing hardware is shown in Fig. 14.5.

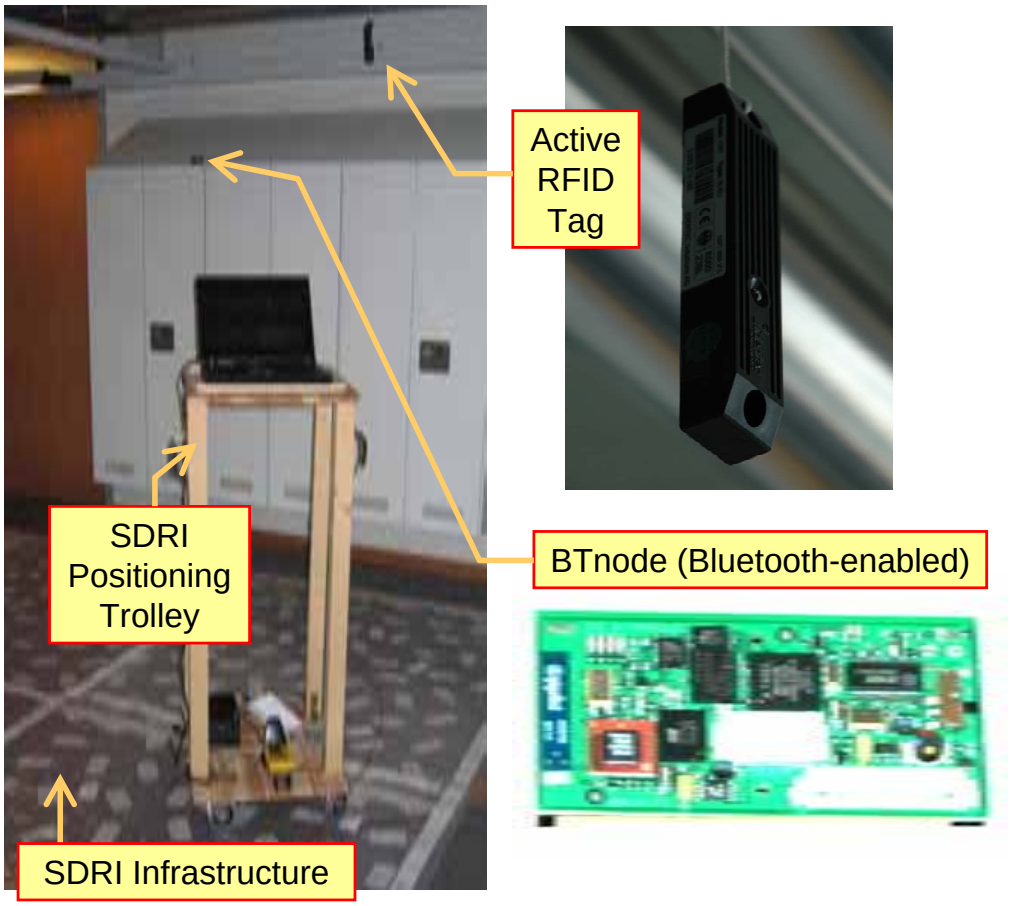

Figure 14.5.: Passive location sensing infrastructure: active RFID tags, active Bluetooth enabled sensor nodes (BTnodes), and densely distributed RFID tags represented by our super-distributed RFID tag infrastructure (SDRI) prototype (see also Sect. 13.3.1)

14.6.2. Active Location Sensing Infrastructure

The *active sensing infrastructure* was represented by two RFID gates. In our setting, an *RFID gate* consisted of an RFID antenna connected to an RFID reader device, which was controlled by an RFID gate application running on a notebook computer. Whenever the RFID gate application detected the presence of a passive RFID tag in the field of the RFID antenna it monitored, it created a time-stamped RFID-gate-event containing the symbolic location identifier (or alternatively the geographic location) of the RFID gate and the ID of the detected RFID tag. The RFID-gate-event was then forwarded to a tuple space managed by a centralized

server that provided a socket-based communication interface. Clients were able to connect to this interface in order to retrieve RFID-Gate-events for a given RFID tag ID.

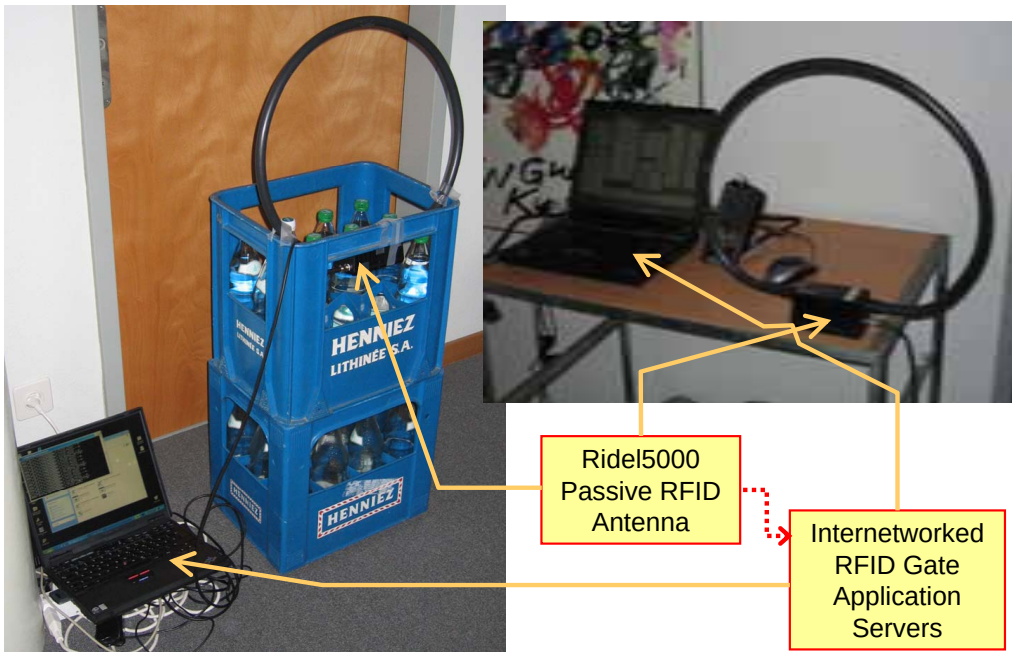

Figure 14.6.: Active location sensing infrastructure: two prototypical RFID gates consisting of (1) RFID antenna, (2) RFID reader, and (3) RFID gate software running on a notebook computer

For each RFID gate, we employed a RIDEL5000-I RFID long range reader in combination with an ANTMR5000 medium range loop antenna by Softronica S.A. Each RFID gate was connected to a notebook computer with Wireless LAN connectivity, on which the RFID gate software was executed (see Fig. 14.6). As smart badges we used Philips I·CODE RFID tags (Type 1) [Phi06].

14.6.3. iPOS Client

On the client side, the iPAQ PDA executing the positioning system was equipped with an i-Card3 PCMCIA RFID Reader by Identec Solutions, to enable the detection of the active RFID beacons. Further, the PDA was connected to a BTnode, which it employed for discovering nearby BTnodes in the environment (using the Bluetooth discovery procedure). Figure 14.7 shows the client hardware setup. Further, for the interaction with the SDRI infrastructure, we used the SDRI Positioning System described in Sect. 13.4.3 as a reference positioning service. The iPOS positioning software on the PDA wirelessly connected to the SDRI positioning service, which was executed on a separate notebook computer mounted on the trolley of the SDRI Positioning System (see Fig. 14.5). The iPOS positioning software was implemented in Java. It was executed on the iPAQ device with the help of the CrE-Me 4.0 Java Virtual Machine by NSICOM, which supports a broad range of Java classes and libraries as of JDK Version 1.3.1.

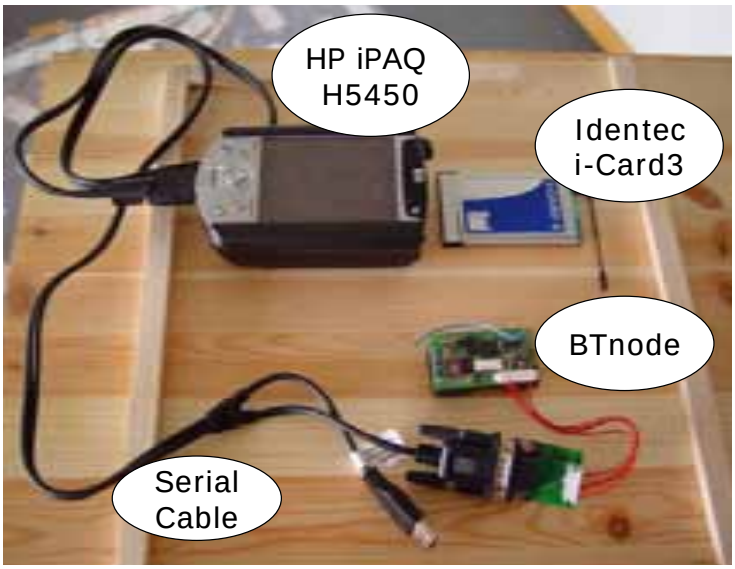

Figure 14.7.: Mobile iPOS client (MoD) with i-Card3 PCMCIA reader and BTnode connected by serial cable

14.6.4. Implemented Sensor Plugins

We implemented the sensor plugin components as independent, active modules that are managed by the resource manager component. In the following, we list the sensor plugins we developed and implemented based on the available sensing technologies. An overview of the sensor plugins and their configurations is shown in Table 14.3.

Sensor	Type	Range/Accuracy	Blocking
Active RFID Beacons	LV	±3 m	yes
Bluetooth Beacons	LV	±30 m	yes
SDRI Prototype	LV	±0.3 m	yes
RFID Gate	LV	±0.5 m	yes
Dead-Reckoning	LV	±1.0 m	yes
Generic LV-Plugin	LV	*	*
Active RFID RSS [2-m-grid]	MC	±5 m	yes
Bluetooth RSS [2-m-grid]	MC	±5 m	yes
WLAN RSS	MC	±10 m	no
Generic MC-Plugin	MC	*	*

Table 14.3.: Overview of implemented sensor plugins. The entries marked with an asterisk depend on the characteristics of the used sensor hardware or third-party positioning service, respectively

Las-Vegas Sensor Plugins

LV-plugins are designed and configured in such a way that they generate sensor events that are considered reliable in the following sense: it is guaranteed that the user's MoD is *always* situated within the areas of influence of the LV-events.

Active RFID LV-Plugin: This LV-plugin treats active RFID tags as individual radio beacons. For every detected active RFID tag, the plugin creates an LV-event containing the ID of the tag as symbolic location identifier. Consequently, each such beacon has to be inserted as an object into the map model at the cell that corresponds to the physical location of the beacon, to enable the fusion algorithm to resolve the geographic positions of the sensor events during the fusion process. The range of the plugin depends on the transmission power of the tags and on the sensitivity of the reader, both of which were configurable in our hardware. We used a configuration that limited the transmission range of the active RFID tags to below 3 m. That means the positioning accuracy per beacon was better than ±3 m. The radio signals of the active RFID tags we used were blocked by walls. Therefore the sensor events were configured as *blocking*.

Bluetooth LV-Plugin: The functionality of the Bluetooth-based LV-plugin is the same as for the active RFID-based LV-plugin: each Bluetooth sender (here: BTnode) is treated as a radio beacon and its Bluetooth MAC address used as symbolic location identifier. Again, each Bluetooth beacon has to be inserted into the map model. The Bluetooth radio signals of our BTnodes were blocked by walls. Therefore the corresponding sensor events were configured as *blocking*. A reliable upper bound of the transmission range of the BTnodes was experimentally evaluated to 30 m and set accordingly in the corresponding sensor event class. As a result, the accuracy of ±30 m of the Bluetooth beacons was much lower than the one of the active RFID beacons.

SDRI Positioning LV-Plugin: We also created an LV-plugin that makes use of the SDRI Positioning System described earlier in Sect. 13.4.3, which directly provides location information in the form of geographic position coordinates at a rate of approx. 1 Hz. The SDRI Positioning LV-Plugin connected to the SDRI Positioning service via a wireless network connection and created an LV-event for each position obtained from the SDRI Positioning service. Further, based on the given hardware configuration and tag distribution density of the prototypical SDRI, the range of the LV-plugin was set to 0.15 m, and the sensor events were configured as *blocking*.

Note that while the SDRI Positioning System currently consists of a considerably large trolley equipped with RFID equipment and a notebook computer, the long-term goal is to make the hardware smaller and wearable. For instance, we envision to integrate the RFID antenna and reader in the user's shoes and to transmit the scanned RFID data wirelessly to the user's PDA for further processing (i.e., positioning).

RFID Gate LV-Plugin: The RFID Gate LV-plugin is the only plugin we implemented that makes use of a background service infrastructure: the LV-plugin has to connect to a central server and ask for any recent RFID-gate-event that contains the user's RFID badge ID as identifier (see Sect. 14.6.2 and Fig. 14.6). For our experiments, we placed the RFID gates such that they covered the area of a single cell of 1 m^2 in the map model, which corresponds to a range of approx. 0.5 m of the respective sensor events. Since RFID

tags cannot be detected through walls, the sensor events were configured as *blocking*.

Dead-Reckoning LV-Plugin: The trolley that carried the SDRI Positioning System (see Sect. 13.4.3) was also equipped with a set of rotation sensors along the middle axis. These sensors were monitored by a dead-reckoning service which, starting from a known initial position, could estimate the current position based on the rotation sensor readings. The dead-reckoning service featured an accumulated drift of below 2% with respect to the driven distance, given a slow walking speed and a straight trajectory. We implemented an LV-plugin based on this dead-reckoning service, setting the range to 1.0 m to account for the drift assuming a test track of approx. 40 m length. The sensor events were configured as *blocking*. However, during the practical experiments, the dead-reckoning hardware based on low-cost LEGO Mindstorms technology proved to be to fragile and susceptible to mechanical defects in the face of several sharp 90-degree turns and a comparably high walking speed. Therefore we decided not to rely on the Dead-Reckoning LV-Plugin during the practical evaluation of the iPOS system.

Generic LV-Plugin: By means of the Generic LV-Plugin, arbitrary third-party positioning services that exhibit Las-Vegas characteristics (see Sect. 14.3.5) can be easily integrated with the iPOS system. For that, a new Generic LV-Plugin instance is created, and its class-specific range and blocking property (see sensor event model in Sect. 14.3.4) are set according to the characteristics of the third-party positioning service. The thus obtained LV-plugin encapsulates any location information obtained from the third party positioning service into the internal iPOS sensor event format, which then can be processed by the fusion engine.

Monte-Carlo Sensor Plugins

MC-plugins generate sensor events that are possibly erroneous with regard to the location information they carry. This means that the user's MoD is *not necessarily* situated within the area of influence of MC-events.

Abstract RSS MC-Plugin: We implemented an abstract MC-plugin based on received signal strength (RSS) measurements of radio signals. The basic mode of operation is as follows: In a *learning mode*, a number of reference positions are learned and stored in the map model using unique symbolic identifiers. Each reference position corresponds to a vector of signal strength tuples of the kind <*sender*, *received signal strength*> (RSS vector). Later, in *positioning mode*, the MC-plugin first measures the signal strength values of all radio transmitters that are received at the current physical location. Then, in a second step, the MC-plugin compares the obtained RSS vector with the RSS vectors stored in the previously built database of reference positions. With the help of a least-squares-metric, the reference position whose RSS vector exhibits the minimum error compared to the current RSS vector is chosen as the current position of the MoD. Obviously, the quality of the positioning depends on the quality of the reference positions. In particular, if no reference position close to the true current position of the MoD is available, or

if the RSS measurements are distorted, the reference position chosen by the MC-plugin as best match for the current position is liable to be wrong or far off the mark. For this reason, this type of plugin is considered unreliable.

The abstract RSS MC-plugin cannot be used directly – instead, it serves as a template for the implementation of RSS plugins based on a particular radio technology. For the creation of a working plugin, only the method that performs the RSS measurements has to be implemented.

Active RFID RSS MC-Plugin: This MC-plugin is an instantiation of the Abstract RSS Plugin. The implementation of the method performing the RSS measurements is based on a native C++ library for accessing the I-Card3 PCMCIA reader, which was integrated via the Java Native Interface. The range of the plugin depends on the transmission power of the tags and on the sensitivity of the reader, both of which were configurable. To be able to receive multiple tags per measurement, we used a tag configuration that, as part of an experimental evaluation, resulted in a range of up to 5 m of the plugin (given a grid-like distribution of the tags with 2 m distance between adjacent tags in the test area). The radio signals of the active RFID tags we used were blocked by walls. Therefore the sensor events were configured as *blocking*.

Bluetooth RSS MC-Plugin: This MC-plugin is an instantiation of the Abstract RSS Plugin. For performing the RSS measurements, the plugin communicates with the connected BTnode via a simple protocol using the serial port. On the BTnode, we used the standard `bt-cmd` utility for performing the Bluetooth discovery. The radio signals of the BTnodes were blocked by walls. Therefore the sensor events were configured as *blocking*. We further determined a range of approx. 5 m (given a grid-like distribution of the tags with 2 m distance between adjacent tags in the test area).

Wireless LAN RSS MC-Plugin: As part of the original positioning system, we also had implemented and tested a sensor plugin based on Wireless LAN RSS measurements [PS03]. The plugin featured an accuracy of approx. 10 m. Since the radio signals of the Wireless LAN access points (i.e., the radio transmitters) reached through walls, the sensor events were configured as *non-blocking*. However, due to a lack of suitable device drivers, we could not access the signal strength values of the Wireless LAN interface on our iPAQ handheld device. As a result, we were not able to port this plugin to the PocketPC platform and use it during the practical experiments we describe in Sect. 14.7.

Generic MC-Plugin: By means of the Generic MC-Plugin, arbitrary third-party positioning services that exhibit Monte-Carlo characteristics (see Sect. 14.3.5) can be easily integrated with the iPOS system. For that, a new Generic MC-Plugin instance is created, and its class-specific range and blocking property (see sensor event model in Sect. 14.3.4) are set according to the characteristics of the third-party positioning service. The thus obtained MC-plugin encapsulates any location information obtained from the third party positioning service into the internal iPOS sensor event format, which then can be processed by the fusion engine.

Auto-Insertion of LV-Event-Objects into Map Models

The availability of self-describing LV-events (i.e., Las-Vegas sensor events that contain their own physical position) can be exploited for the *self-organization* of map models: the system can be configured to automatically create a suitable map object and insert it into the map model upon the first detection of a self-describing LV-event. Such an *auto-insertion* or *auto-mapping* would significantly reduce the maintenance overhead caused by the need of manually inserting LV-event-objects into the corresponding map models. However, this feature has not been implemented yet in the current version of the iPOS prototype system.

14.6.5. Management Tools

To facilitate the creation and maintenance of map models, we developed a graphical map builder and topology editor that allows the user to create or edit maps. The topology editor also allows the user to place and name objects in the map, to draw walls, and to insert transition cells that link a map to another map (Fig. 14.8).

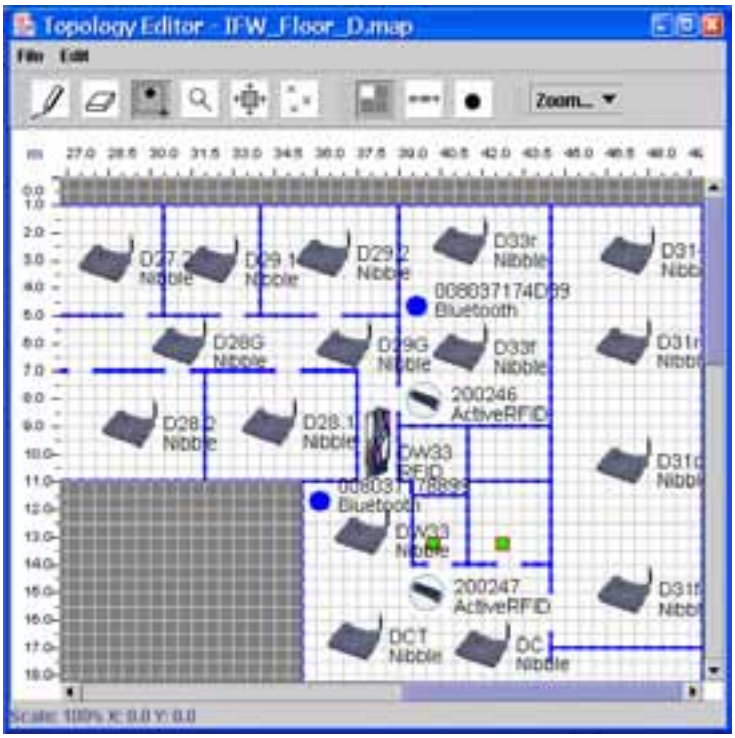

Figure 14.8.: Snapshot of the topology editor application

The map builder has been enhanced by a simulation mode that allows the manual *emulation of sensor events*, which proved suitable for preliminary testing of new heuristics and algorithms for positioning. We have also realized a monitor application that allows to continuously track the position of a MoD, mark its position on the associated map, and visualize the current occupancy probability distribution in the grid map (Fig. 14.9).

The functioning of the management tools is fully decoupled from the operation of the positioning system – they may resided anywhere in the network. This permits to run only those components on a resource-limited MoD which are indispensable for positioning.

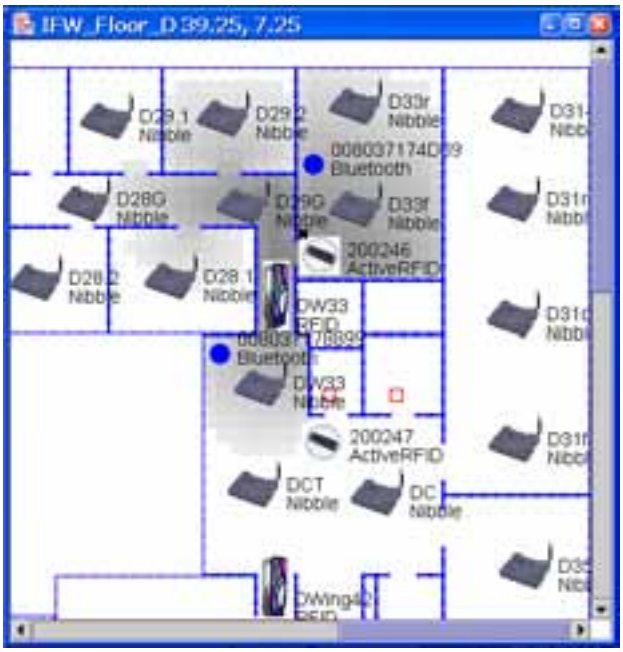

Figure 14.9.: Snapshot of the monitor application

14.7. Experimental Evaluation

14.7.1. Experimental Setup

In preparation of our practical experiments, we deployed our hardware for the sensing infrastructure in the corridor of our office building at ETH Zurich. We also created a grid map model of the corridor with square cells of 1 m^2 size.

The BTnodes serving as Bluetooth senders and the active RFID tags were distributed in a grid with 2 m distance between adjacent entities, with a displacement of 1 m between the two grids. The map model was updated by inserting BTnode- and active-RFID-objects into the map at the respective positions as shown in Fig. 14.10. An area of the corridor was also covered with densely distributed RFID tags (see area shaded in grey in the figure), using our RFID-tagged SDRI foil templates described in Sect. 13.3.1. In the figure, we can also see the locations of the map objects placed for the two RFID gates (i.e., the areas covered by their antenna fields), marked with grey rhombuses (diamonds) in the grid map model. The map objects for the reference positions that were learned for the Bluetooth RSS MC-Plugin are marked with small squares in the map model. We further laid out a test track of 38 m, which started in a side hall and led in a loop through the prepared corridor as displayed in Fig. 14.10. Images of the actual deployment of the hardware infrastructure are shown in Fig. 14.5 and Fig. 14.6 in Sect. 14.6.

We further developed a *sampling tool* based on the map builder application to assist us in performing the practical measurements. It operates as follows: by clicking onto a cell in the displayed map model, a time-stamped entry is stored in a log file containing the true position coordinates of the cell according to the map model, the current position estimate calculated by the iPOS positioning system, and the deviation between the two position values (see Fig. 14.11). During the experiments, the sampling tool was executed on a separate notebook computer, so that it could be operated by a second person different from the person carrying

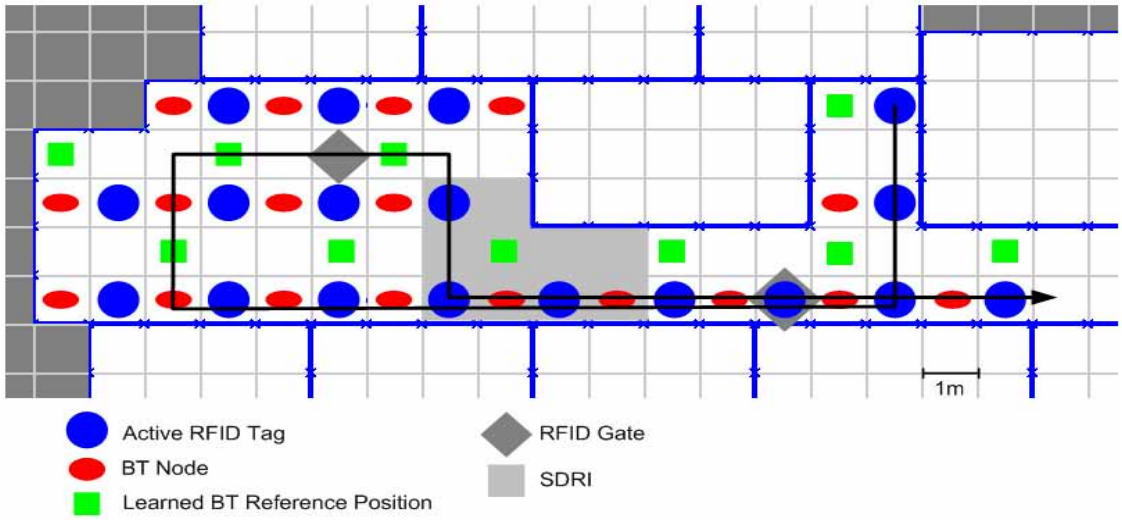

Figure 14.10.: Excerpt of the grid map model showing the experimental setup and test track

the MoD executing the iPOS application. The sampling tool retrieves the position estimates from the iPOS positioning server running on the MoD via a TCP/IP over Wireless LAN connection.

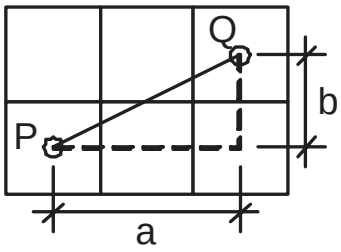

Figure 14.11.: Calculation of the positioning error e_{pos} between a reference position P and the corresponding position Q estimated by the iPOS system: $e_{pos} := \sqrt{a^2 + b^2}$. The geographic position coordinates of P and Q are defined by the center point coordinates of the respective cells in the grid map model

14.7.2. Experimental Method

The PDA running the iPOS application was put on top of the trolley that also carried the SDRI Positioning System. One person pushed the trolley along the test track marked in Fig. 14.10, while a second person initiated the samples at well-defined locations. For the synchronization of the sampling with the trolley movement, we used a metronome, which set a 1 Hz audible beat.

We performed experimental measurements based on two different ways of traversing the test track:

(A) *Stop-And-Go Mode:* The person pushing the trolley advances one meter (i.e., one cell in the model) every five seconds, by walking for two seconds to the

next position and waiting for three seconds before moving on. Each waiting position corresponded to a cell in the map model, which in practice was achieved by using the regularly deployed active RFID and BTnode beacons as indicators for the center points of the respective cells. At the end of each waiting phase, the second person initiates a sample for the current position using the sampling tool described in Sect. 14.7.1.

(B) *Continuous Mode:* The person pushing the trolley advances at a constant speed of 0.4 m/s, while the second person takes position samples with the sampling tool after every two meters of distance the trolley has covered.

The precision of position estimates of the iPOS system is defined by the cell size in the grid map model, which was 1 m^2 in our experimental setting. Therefore, the inherent average position error e_{avg} of position estimates was approx. 0.35 m, given that the iPOS system returns the coordinates of the center point of the cell in which the MoD is thought to be located (Eqn. 14.11, with $x_{avg} = y_{avg} = \frac{1}{2} * \frac{1}{2}$ m). The inherent maximum position error e_{max} is 0.71 m, in case the true position is situated in one of the corners at the maximum distance from the center point of the cell (Eqn. 14.12, with $x_{max} = y_{max} = \frac{1}{2}$ m).

$$e_{avg} := \sqrt{{x_{avg}}^2 * {y_{avg}}^2} = \sqrt{(\frac{1}{2} * \frac{1}{2} \text{ m})^2 * (\frac{1}{2} * \frac{1}{2} \text{ m})^2} = \frac{1}{4} * \sqrt{2} \text{ m} \approx 0.35 \text{ m} \quad (14.11)$$

$$e_{max} := \sqrt{{x_{max}}^2 * {y_{max}}^2} = \sqrt{(\frac{1}{2} \text{ m})^2 * (\frac{1}{2} \text{ m})^2} = \frac{1}{2} * \sqrt{2} \text{ m} \approx 0.71 \text{ m} \quad (14.12)$$

This means that the *precision* of the iPOS positioning system based on our experimental setting was 0.35 m in 50% and 0.71 m in 100% of all position estimates.

14.7.3. Experimental Results

We performed six positioning experiments based on different combinations of sensor plugins. An overview of the experimental configurations and the obtained positioning results is displayed in Table 14.4. For the experiments in stop-and-go mode (mode A), we performed five test runs per configuration, collecting 39 samples per run, which resulted in an overall number of 195 samples. For the experiments in continuous mode (mode B), five test runs were performed and 20 samples were collected during each run, yielding a total of 100 samples per configuration. The SDRI Positioning LV-plugin and the RFID Gate LV-plugin were modeled with the accuracy of one cell each. The Active RFID LV-Plugin was modeled with an accuracy of five meters.

The detailed test results for each experiment are shown in Figures 14.12–14.17. Each figure shows the minimum, average, and maximum positioning error for each positioning sample taken along the test track during the respective experiment – apart from Fig. 14.17, which only displays data obtained from a single test run.

14.7.4. Interpretation

By comparing experiments number 1 and 2 based on the same configuration of LV-plugins, we can see that the accuracy of the iPOS positioning service in the

Exp.	Config.	Active Plugins	Mode	#Runs	n	α [m]	σ [m]
1	1	Active RFID LV RFID Gate LV SDRI Pos. LV	A	5	195	1.14	0.97
2	1	Active RFID LV RFID Gate LV SDRI Pos. LV	B	5	100	1.64	1.14
3	2	Active RFID LV	A	5	195	1.44	1.01
4	2	Active RFID LV	B	5	100	3.03	2.80
5	3	Active RFID LV RFID Gate LV SDRI Pos. LV Bluetooth RSS MC	A	5	195	1.25	1.25
6	4	Bluetooth RSS MC	A	1	39	6.66	3.62

Table 14.4.: Overview of experiments based on different combinations of sensor plugins. For each experiment, n describes the number of collected samples, α the average error in meters, and σ the standard deviation of the error in meters

stop-and-go mode is 30% higher compared to the continuous mode. With regard to the map model, the accuracy of 1.14 m with a standard deviation of about 1 m for the stop-and-go mode can be interpreted as follows: the average error amounts to approx. one cell in horizontal or vertical direction, with a standard deviation of one cell. Likewise, the error for the continuous mode in experiment no. 2 can be interpreted as amounting to approx. one cell in diagonal direction of the correct cell in the grid, with a standard deviation of approx. one cell.

We found that the main reason for the decline in accuracy with increased mobility was the short delay between the creation of the sensor event and the processing in the fusion engine, which in particular affected the location information obtained via wireless connection from the SDRI Positioning LV-plugin and from the RFID Gate LV-plugin. We can see in Fig. 14.12 and 14.13 that the average error approximately doubles at those places along the path where the SDRI Positioning and RFID Gate plugins were encountered by the MoD.

This observation also holds true for the results obtained using only the Active RFID LV-plugin for positioning (Fig. 14.14 and 14.15): the average error of the continuous experiment (experiment no. 4) increased by approx. 110% in comparison to the stop-and-go experiment (experiment no. 3), with an even higher increase of the standard deviation.

The stop-and-go experiment based on the active RFID tags modeled as a reliable LV-plugin further shows that the average and maximum accuracy of the majority of position estimates calculated by the iPOS fusion engine were contained in the range of the corresponding LV-plugin. The reason for the outliers around the distance of 11 m and 32 m we identified as timing problems due to temporary delays in the event processing and fusion procedure on the resource-limited mobile iPAQ device. The timing problem should be addressed by introducing an explicit task scheduling, or by increasing the safety margin of the ranges defined for the LV-plugins.

Comparing the performance of the Active RFID LV-plugin in experiments no. 3

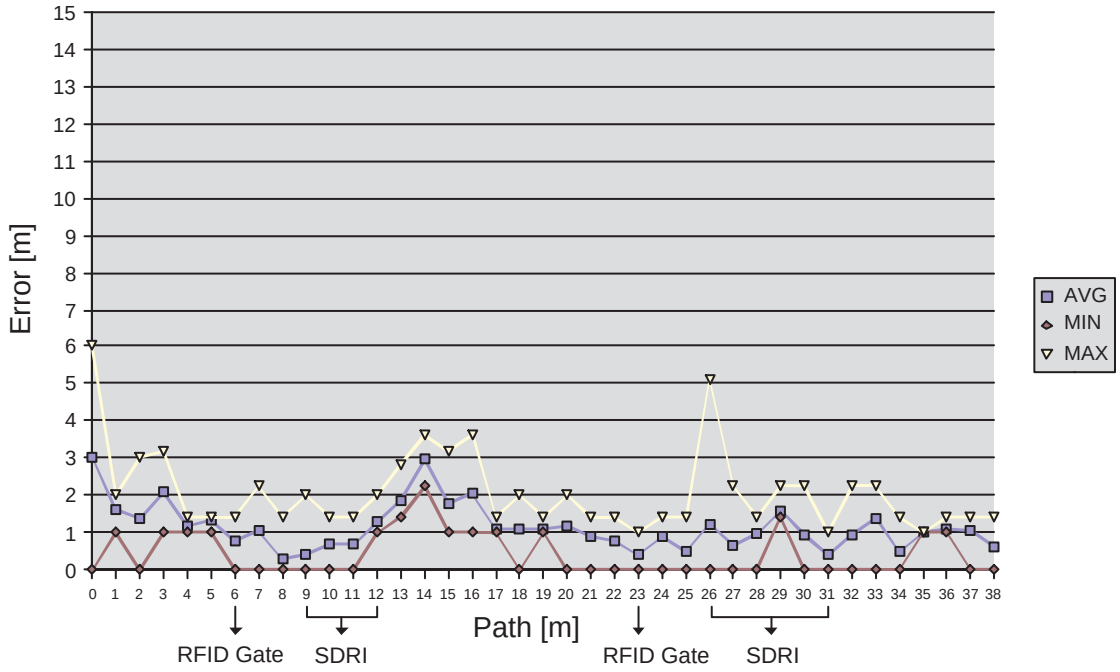

Figure 14.12.: Positioning errors for experiment no. 1, configuration no. 1, mode A. Plugins: Active RFID LV-plugin, RFID Gate LV-plugin, and SDRI Positioning LV-plugin ($runs = 5$, $n = 195$, $\alpha = 1.14$ m, $\sigma = 0.97$ m)

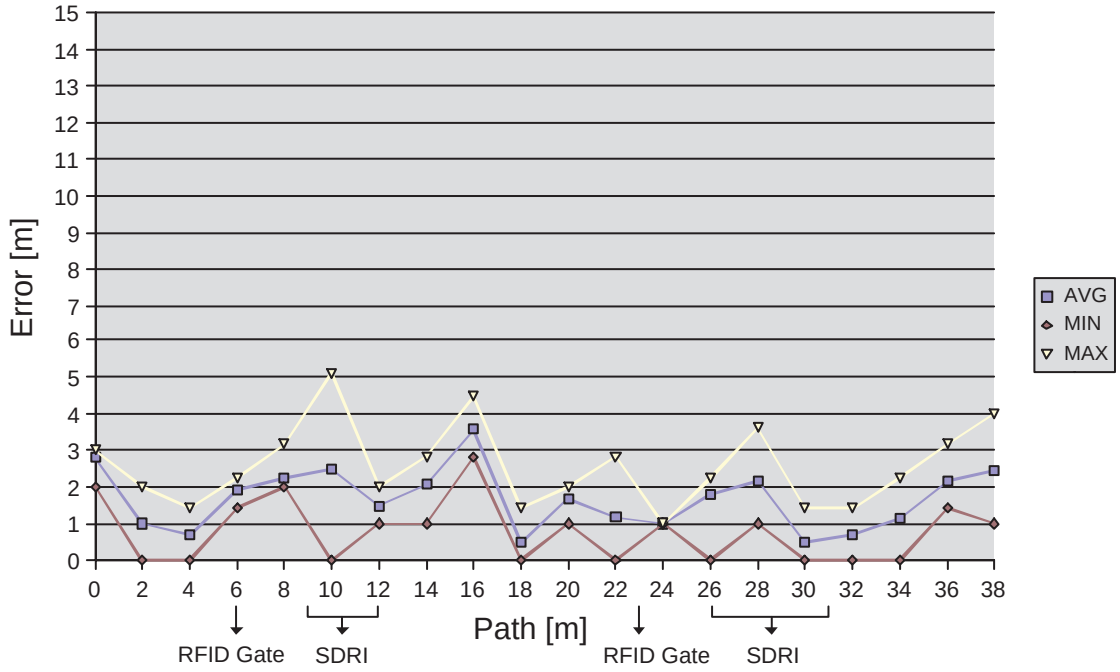

Figure 14.13.: Positioning errors for experiment no. 2, configuration no. 1, mode B. Plugins: Active RFID LV-plugin, Bluetooth LV-plugin, RFID Gate LV-plugin, and SDRI Positioning LV-plugin ($runs = 5$, $n = 100$, $\alpha = 1.64$ m, $\sigma = 1.14$ m)

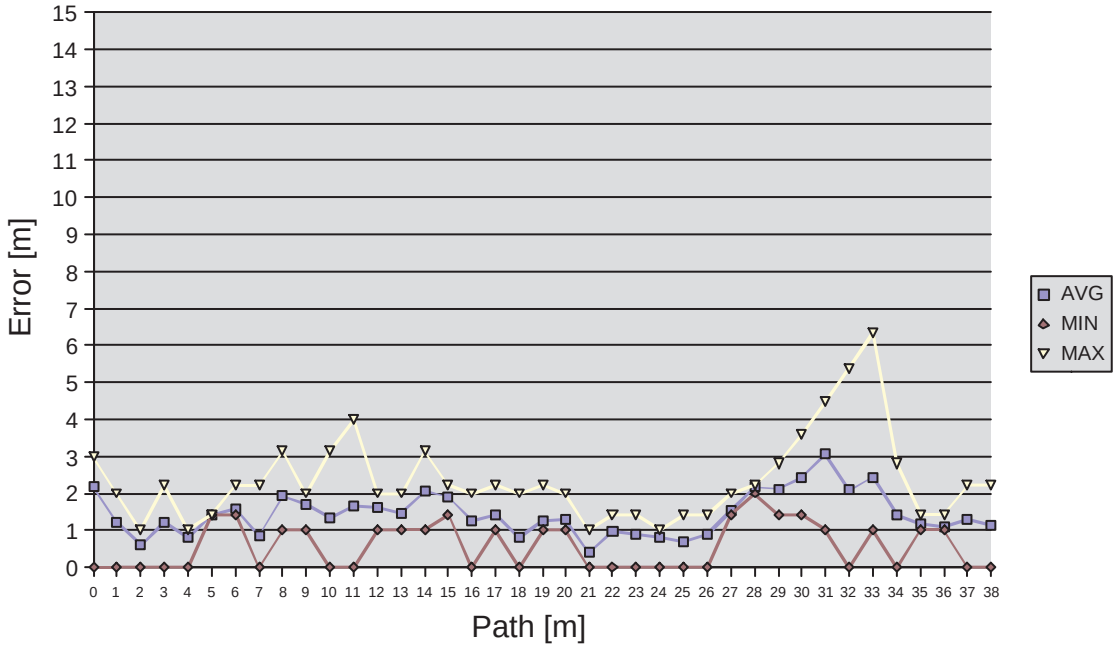

Figure 14.14.: Positioning errors for experiment no. 3, configuration no. 2, mode A. Plugins: Active RFID LV-plugin ($runs = 5$, $n = 195$, $\alpha = 1.44$ m, $\sigma = 1.01$ m)

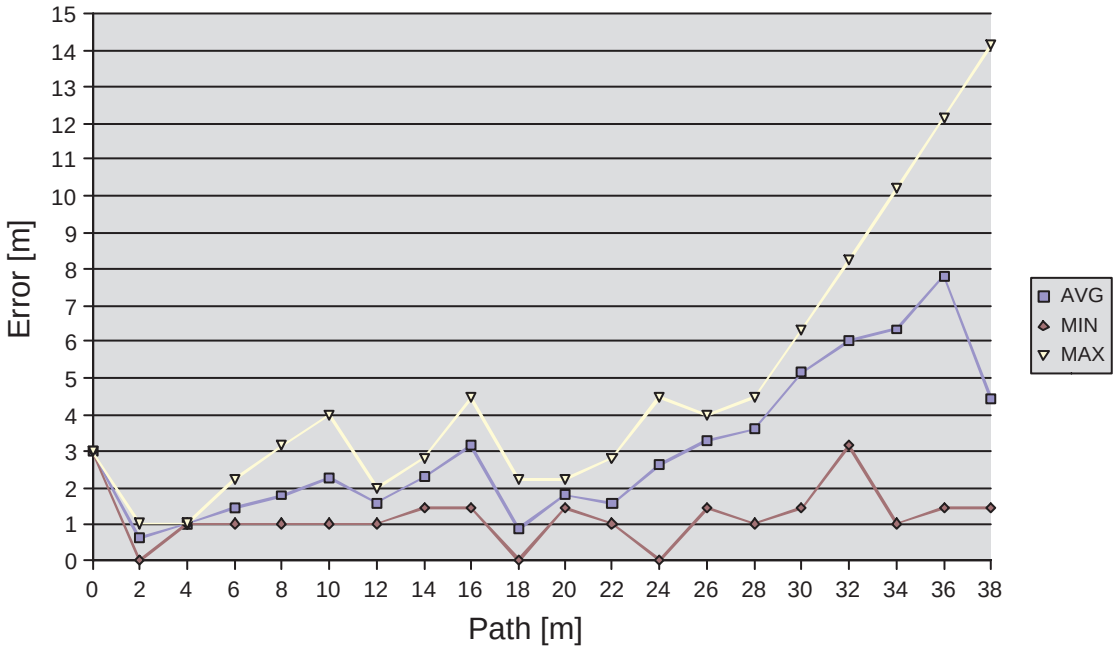

Figure 14.15.: Positioning errors for experiment no. 4, configuration no. 2, mode B. Plugins: Active RFID LV-plugin ($runs = 5$, $n = 195$, $\alpha = 3.03$ m, $\sigma = 2.80$ m)

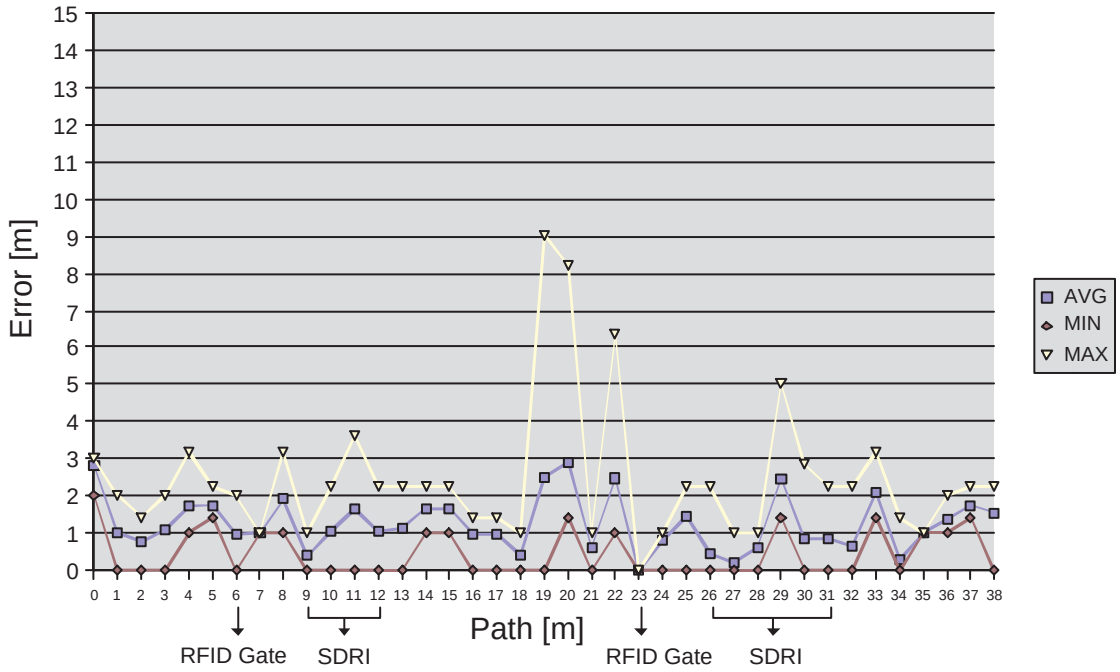

Figure 14.16.: Positioning errors for experiment no. 5, configuration no. 3, mode A. Plugins: Active RFID LV-plugin, Bluetooth LV-plugin, RFID Gate LV-plugin, SDRI Positioning LV-plugin, and Bluetooth RSS MC-plugin ($runs = 5$, $n = 195$, $\alpha = 1.25$ m, $\sigma = 1.25$ m)

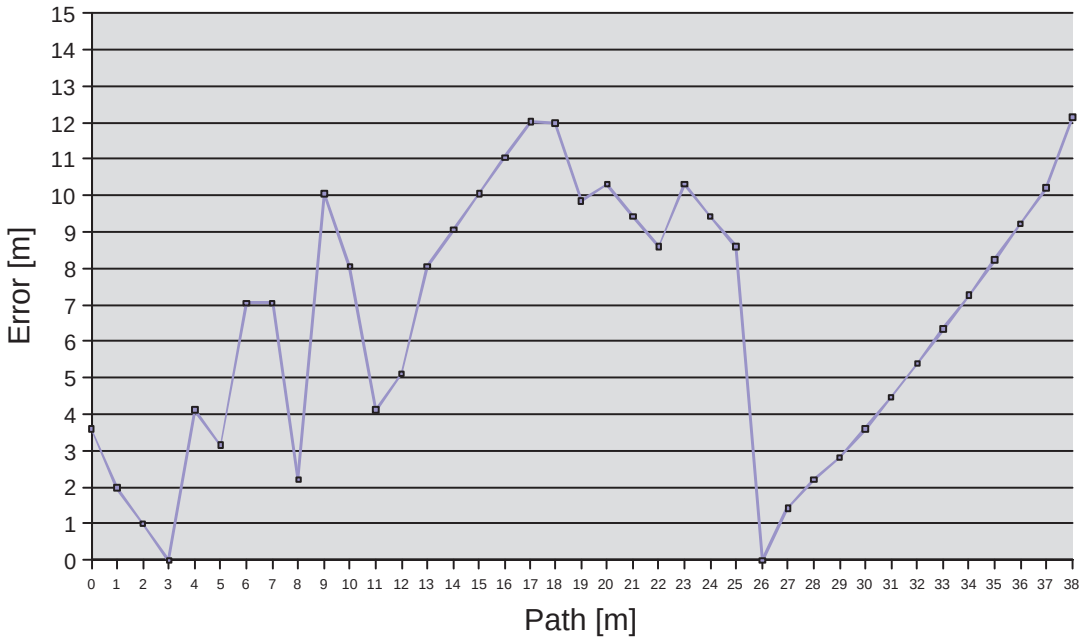

Figure 14.17.: Positioning errors for experiment no. 6, configuration no. 4, mode A. Plugins: Bluetooth RSS MC-plugin ($runs = 1$, $n = 39$, $\alpha = 6.66$ m, $\sigma = 3.62$ m)

and 4 with the overall performance of the multi-plugin experiments no. 1 and 2, we can see that the accuracy of the fusion process is indeed determined by the accuracy of the most accurate active LV-plugin at a time. In our case, this concerned the stretches along the test track where the areas of influence of the RFID Gate LV-plugin and of the SDRI Positioning LV-plugin were entered by the MoD.

Finally, we performed an experiment where we added the Bluetooth RSS Plugin to the plugins of configuration 1 (experiment no. 5, configuration no. 3). Here we can see that the overall accuracy has slightly decreased in the average case, with a few outliers with regard to the maximum positioning error. The reason for this effect was that the Bluetooth RSS MC-plugin on its own proved to be very unreliable and inaccurate, as the exemplary positioning results of experiment no. 6 displayed in Fig. 14.17 show. And this unreliability of the MC-plugin had a pronounced negative effect whenever only MC-events and no LV-events were detected, as the Clip was then recalculated based on the potentially very inaccurate MC-events. In the experiment based on LV-plugins only, the Clip was only extended according to the mobility heuristic and not changed completely in the absence of LV-events. To address this issue, we decided to change the Clip calculation procedure in the case that only MC-events are obtained to *joining* the existing *Clip* with the Regions of the MC-events rather than *replacing* the *Clip* with the union of the *Regions* of the MC-events (see Sect. 14.4.1).

Initially we also planned to perform experiments using the Active RFID RSS MC-Plugin in addition to the plugins of configuration 1. However, this was technically infeasible since the Active RFID LV-Plugin and Active RFID RSS MC-Plugin could not be employed in parallel on the MoD: both plugins make use of the same RFID antenna hardware and driver and required incompatible settings concerning the sensitivity and range of the antenna field.

14.8. Discussion

In the following, we discuss our experimental results according to the taxonomy for assessing location technology proposed by Hightower [HB01], which considers scalability, cost, recognition and limitations. We further assess the performance and dependability of the system, and suggest a number of improvements as part of future work.

14.8.1. Scalability

A major advantage of the iPOS positioning system is its *extensibility*, which is achieved by means of an *open plugin architecture* and the support of global positioning coordinates according to the WGS-84 standard [Eur06]. This makes it possible to integrate arbitrary location sensing technologies and third-party positioning services alike with little effort. For that reason, the iPOS system scales well in terms of *versatility* and variety of supported location sensing technologies.

Owing to the extensibility of the system architecture, the iPOS system is in a position to exploit a large number of sensor technologies and sensing infrastructures as they are present in ubiquitous computing infrastructures. For instance, the system can interface existing positioning services as well as extract location information from wireless network infrastructures, Bluetooth hotspots, RFID-tagged

carpets (e.g., [Vor05]), or stationary sensor networks. The iPOS system therefore also scales well with regard to *cost* as a factor.

The iPOS system also scales well with respect to the *number of sensor plugins* that can be operated in parallel, despite the application of a grid-based map model. The main limiting factor for the number of supported active plugins is the amount of available system resources on the MoD, as the number of cells that have to be processed in the map-assisted fusion engine is constant in the presence of LV-events, and increases linearly with the number of MC-events (or MC-plugins) that have to be processed in case no LV-events are available. Further, in the worst case, the number of multiplications per cell in the *Clip* increases linearly with the number of obtained sensor events. Besides, in our system the normalization of the occupancy probabilities is performed on the fly during the fusion process and does not pose a performance issue. As a result of the low overhead and complexity, the positioning architecture is particularly suited for the execution on lightweight, resource-limited mobile devices.

Concerning the maintenance of the map model, the system is further capable of performing a form of *self-organization*: if an LV-plugin creates an LV-event that already contains geographic position information in addition to its symbolic location identifier, a corresponding LV-object can be automatically inserted into the map model. This has two advantages. Firstly, the newly mapped LV-event can now be considered during the *negative feedback* calculation in the fusion engine (see Sect. 14.4.2). Secondly, the *Region* of the mapped LV-event can be precalculated and stored with the map model, increasing the efficiency of the fusion procedure on the resource-limited mobile device during runtime.

Finally, the iPOS plugin architecture scales well with regard to the *physical distribution of sensing hardware*. Physical or logical sensors can be distributed across various devices, as long as these sensors feature a socket-based interface and network connectivity. Then a sensor plugin can be created for the iPOS system that accesses the external sensor by means of a wireless network connection. Thus resource-limited devices can outsource sensor hardware and remain in a position to perform robust positioning based on multi-sensor fusion. For instance, in our iPOS prototype system, the SDRI Positioning LV-plugin executed on the MoD connected wirelessly to an external computing device that ran the SDRI Positioning service and controlled the RFID reader/antenna hardware.

14.8.2. Cost

Principally, the iPOS system can be configured to make use of any readily available hardware (on the mobile client executing the iPOS system) or infrastructure (in the user's environment) that can be exploited for location sensing and positioning. Therefore, in general, the usage of the iPOS system does not result in extra costs. For instance, a PDA featuring Wireless LAN and Bluetooth connectivity can use these available radio interfaces for location sensing. The purchase of additional location sensing hardware, such as a mobile RFID reader or a GPS receiver module, for instance, may be considered if the needs of the user require it and/or if the physical support infrastructure is already available (e.g., a grid-like or random distribution of stationary radio beacons as part of a sensor network).

14.8.3. Recognition

Our experimental results indicate a good accuracy of the fusion-based positioning system in comparison to the accuracy of the individual sensing technologies.

Based on our developed fusion algorithm and the explicit modeling of reliable sensor events, the iPOS system further is capable of providing *quality-of-service (QoS) guarantees*. Firstly, the fusion engine guarantees that overall accuracy of the fusion process is at least as good as the accuracy of the most accurate LV-event that was part of the input during one iteration. Secondly, given that the LV-plugins were modeled with care and the positioning system does not suffer from timing or synchronization problems, the iPOS system can provide QoS guarantees with regard to the achieved accuracy if at least one LV-plugin generates sensor events: the position of the mobile device is guaranteed to be within the area that corresponds to the current *Clip* in the map model. This knowledge can be used to inform applications using the positioning service about the currently guaranteed quality of service, that is about the currently guaranteed accuracy in case that LV-events are available, or about the lack of such QoS guarantees if only unreliable MC-events are fed into the fusion engine of the iPOS system.

14.8.4. Limitations

Time Synchronization and Task Scheduling

During our experiments, we recognized *time synchronization* as an important challenge with regard to the processing of sensor events. If the clocks of the computers involved in the generation of sensor events are out of sync, it can happen that the fusion engine dismisses new sensor events as too old or considers outdated sensor events to be up-to-date. This can result in a degraded quality-of-service and lower dependability of the probabilistic fusion process. A possible solution is to employ explicit time synchronization mechanisms and to perform task scheduling for a better coordination of active plugins and the fusion engine. A further possible solution is to increase the ranges of time-critical sensor events to account for a certain acceptable maximum delay. In our case, we manually calibrated the clocks of the different computers involved in the event generation process (i.e., the computers controlling the RFID gates and the SDRI Positioning service) before carrying out the experiment, but this is not a practical solution in real-life application scenarios.

Flexibility of the Grid Map Model

An implementation-specific limitation of our system is the use of a fixed cell size per map. Variable cell sizes would allow a more fine-grained resolution for specific areas of a map if needed. It would also avoid an unnecessarily high cell density in areas where applications are satisfied with less precise position information, thus further reducing memory consumption and computational complexity. Nevertheless, in our system, using fixed cell sizes proved not to be a significant disadvantage: the iPOS system is targeted at existing location sensing technologies and infrastructures that are generally found in ubiquitous computing environments. These location sensing technologies often feature a limited accuracy that is lower than the precision of the iPOS system even when using a relatively large fixed cell size of 1 m $\times$ 1 m or beyond.

14.8.5. Performance

Given our experimental setting and configurations, the mobile handheld we used was in a position to calculate position updates at a rate of up to 1 Hz. We consider this a very good result, especially since a higher update rate was not practical as the maximum response rate of our sensor plugins themselves was approx. 1 Hz or below. A profiling of the iPOS system running on the handheld device showed that the delay caused by the map-based fusion processing was negligible as long as sensor events were obtained regularly (keeping the Clip at a small controllable size).

Only in cases when the mobility heuristic was applied repeatedly for a longer period of time without any sensory input in between, the *Clip* would grow beyond a size where the recalculation of the cell occupancy probabilities negatively affected the position update rate. To address this issue, we suggest to suspend the application of the mobility heuristic and the calculation of position estimates whenever more than I consecutive iterations yield no location information in the form of sensor events, and to resume operation as soon as new sensory input is available. The value of I should be calibrated during practical experiments and set to the value up to which the application of the mobility heuristic does not significantly affect the performance of the iPOS system, which in large parts depends on the computing power of the mobile device. Furthermore, the value of I should not be chosen too high – else the quality of the iPOS position estimates is liable to suffer due to the increase of uncertainty about the user's true position when repeatedly no new location information is obtained from the sensors.

14.8.6. Dependability

Liveness

The iPOS system returns position estimates as long as sensor input is available from the sensor plugins during subsequent runs of the positioning algorithm. In the process, the *Clip* management ensures that the *Clip* only contains cells with an occupancy probability greater than a minimal threshold δ, and that the probabilities of all cells in the *Clip* sum up to one. The *Clip* management further ensures that the Clip is never empty. Consequently, the heuristic for calculating a position estimate will always return an estimate as long as sensor events are available. So the *liveness* of the positioning system is ensured as long as at least one of the sensor plugins is functioning.

Safety

On the one hand, if only unreliable sensor plugins (i.e., Monte-Carlo sensor plugins) provide sensor events, then the iPOS system does not guarantees the correctness of its position estimates with respect to accuracy. In this case, no safety properties are assured.

On the other hand, if reliable sensor events (i.e., Las-Vegas sensor events) are available, the iPOS system provides quality-of-service guarantees with regard to the achieved accuracy, given that the LV-plugins were modeled correctly and the positioning system does not suffer from timing or synchronization problems: then per design the position of the mobile device is guaranteed to be correct, which

means it is within the area that corresponds to the current *Clip* in the map model. The system in this case *satisfies* the safety property that the accuracy of position estimates is guaranteed to be within certain well-defined boundaries.

Adaptability

The resource manager of the iPOS system can dynamically load and unload sensor plugins during runtime. This provides the basic functionality for controlling energy consumption or computational load caused by the active sensor plugins on the MoD. This way it is also possible to adapt the desired degree of positioning accuracy during operation according to the requirements of the clients of the iPOS system.

Fault Tolerance

Due to the probabilistic nature of the positioning algorithm and the high-level sensor fusion applied in the process, the failure of individual sensor plugins leads to a *graceful degradation* of the quality (i.e., accuracy) of the position estimates. Our fusion algorithm is robust in terms of tolerating the failure of single sensor plugins. The absence of sensor plugins only affects the quality of the positioning, but as long as there is some sensory input available, the positioning algorithm stays "alive" and continues to generate position estimates, maximizing the *availability* of the service. If a highly accurate sensor-plugin failed and the overall accuracy of position information dropped from cell level to room level, for instance, applications that are satisfied with rough position estimates would still be fully functional.

The iPOS system further tolerates the temporary absence of all sensory input by applying a mobility heuristic that adjusts the cell occupancy probabilities in the grid model to account for the uncertainty introduced by a potential movement of the MoD (see Sect. 14.4.5).

Additionally, even in case all sensory input temporarily fails to appear, movement heuristics and dead reckoning techniques can be applied to get a rough picture of the location of the object, such as by counting the number of footsteps of a walking person [VMKA02], for example. Alternatively, the momentary absence of all sensory input can be compensated by deploying heuristics for a short-time position prediction.

The system is also resilient to sensors providing inaccurate data. Location information gained from such sensors is mapped to unreliable sensor events that reflect the corresponding uncertainty of the provided location information (see Sect. 14.3.5). Even false sensor data or outliers that occur sporadically (e.g., due to transient interference) are tolerated by the position fusion engine, as the derivation of position estimates is performed with the help of a maximum-likelihood heuristic that considers the reachability of new positions based on a *mobility model* and using the last estimated position as a starting point (see Sect. 14.4.6).

Privacy

In many contexts, location information is considered as sensitive information [HS02]. In our system, concerns about the *privacy* of location information are addressed by focusing on the use of locally acquired sensor information, that is information which is passively available and requires no external requests or service infrastructures.

Thus, by enabling the user to configure the iPOS system to solely rely on local location sensors, the system can be prevented from introducing new channels by which location information about the MoD itself is revealed to the surrounding environment. However, in some cases such a policy is difficult to enforce due to technical restrictions. For instance, a node communicating over Wireless LAN reveals its location (to a certain degree) to the base station, which cannot be prevented by our system.

14.8.7. Future Improvements

The development and implementation of further sensor plugins that make use of "classic" location sensing technologies such as GPS or GSM, which are already truly ubiquitous today, is an important requirement. This is necessary in order to increase the practicality and coverage the iPOS system (e.g., in outdoor scenarios), and to make it more suitable for daily use.

Moreover, in the prototype system, the map handler currently loads new maps from the local file system. Here a mechanism for discovering local map servers and for dynamically loading missing maps should be implemented, especially to cover previously unknown areas for which no maps have been preloaded on the MoD.

With regard to the adaptability of the system, we are envisioning the creation of new sensor plugins on the fly. Based on the generic plugin templates, a plugin could be instantiated and activated automatically during runtime whenever a location sensing hardware or third-party positioning service becomes available. This would require some sort of discovery or notification mechanism for the iPOS system to become aware of such new resources, possibly including a meta description language defining the properties of the respective location sensing technology (e.g., range and blocking property, symbolic identifier/class name for the type of sensor, names of associated maps, and so on). Such a mechanism would enable the iPOS system to create *adapters* for using previously unknown location sensing technologies or services that are encountered by the MoD along the way.

Another interesting option for future work is the porting of the iPOS system onto a mobile smart phone platform, where existing GSM, Bluetooth, and potentially also Wireless LAN and Near Field Communication interfaces could be exploited for location sensing. The increasing proliferation of smart phones makes them a challenging target platform, with a large user base for user studies.

14.9. Conclusion

We have presented a positioning system for ubiquitous computing that emphasizes the robustness and fault tolerance of the core functionality, namely self-localization of a "smart object" or mobile device. This is achieved by means of redundant location sensing and data fusion. The system architecture allows for simple integration of multiple sensors due to the loose coupling between the acquisition of sensory location information and the fusion-based positioning algorithm. The open plugin architecture facilitates the integration of arbitrary location sensing technologies and existing third-party positioning services. Further, by using a grid map model, the iPOS system is capable of integrating both symbolic or geographic representations of position information, including local and global geographic position coordinates.

A special feature of the fusion algorithm is that, by distinguishing and modeling reliable and unreliable sensor plugins, it is able to provide quality-of-service guarantees for positioning results under certain conditions.

Moreover, the iPOS system focuses on using an existing communication and identification infrastructure rather than dedicated hardware for localization. This has the advantage of avoiding additional *cost* for installation and maintenance. Nevertheless, our system can easily be integrated with a dedicated positioning infrastructure (e.g., a super-distributed RFID tag infrastructure). This makes the iPOS system also suitable for environments where certain areas require high location resolution, accuracy and precision. By using a unified internal format for sensor events that abstracts from the sensing mechanisms used in the different sensor plugins, our system supports a seamless transition between areas where different positioning technologies are available, including smooth hand-offs between indoor and outdoor areas.

Finally, as a proof of concept, we presented and experimentally evaluated a prototype implementation of the iPOS system.

14.10. Related Work

14.10.1. Non-Redundant Positioning Systems

An overview of (indoor) positioning systems for ubiquitous environments and their relation to robot positioning is given in [HB01]. There are numerous systems based on dedicated hardware, such as cameras [KHM+00, RS00, dI01], infrared beacons [WHFG92], ultrasonic emitters [WJH97, PCB00], pressure sensors [AJLS97, OA00], and special hardware for dead reckoning [VMKA02]. Some effort has also been invested in systems that require no additional hardware to locate objects. The RADAR project [BP00] draws location information from the signal strength of WLAN installations. In our system, we used basically the same approach for our Bluetooth RSS MC-plugin. Abowd et al. [ABO02] use biometric sensors and RFID for immediate location updates.

14.10.2. Grid-Based Positioning Systems

Grid based positioning is a standard technique for robot positioning [BFH97, FBDT99, FBT99]. While the basic mathematical techniques are applicable also in positioning for ubiquitous computing environments, the goal is quite different. In ubiquitous computing, positioning of people (or their devices) is prevalent, thus different sensors are used, and many applications don't require a high degree of accuracy (an accuracy of approximately one meter is often sufficient, or only symbolic location information is necessary). Further, the traditional grid-based positioning systems are not particularly suited or adapted to be used on small, resource limited mobile devices, as they typically rely on a fully equipped mobile robot with much more powerful processing and sensing capabilities.

14.10.3. Non-Redundant Multi-Sensor Positioning Architectures

Data fusion for positioning in wireless networks is discussed in [KOB01] with an emphasis on lower levels. Since that work is based on measurements such as time of arrival and time difference of arrival, it is not directly applicable to indoor positioning. However, such techniques can be incorporated in our system in order to increase the accuracy of position information drawn from outdoor sensors.

Anne et al. combine the strong identification of RF-based location sensing (RFID and Wi-Fi) with the accuracy of computer vision-based-tracking to provide a new solution for multi-scale and multi-target indoor location sensing [ACDP05]. The system was designed to make use of an existing sensing infrastructure. Its main goal is to support indoor location-based applications that feature strong precision and scalability requirements that usually cannot be met by any single of the location-sensing technology used in the process.

14.10.4. Redundant Multi-Sensor Positioning Architectures

The *Location Stack* model proposed by Hightower et al. in [HBB02] provides a universal, multi-layered design abstraction for location-aware ubiquitous computing systems. It provides a framework which allows the integration of arbitrary fusion techniques. Their practical implementation [GLHB03] is based on Bayesian filtering mechanisms for the data fusion layer, including Kalman- and particle filters [FHL+03]. While Kalman filters are computationally efficient, they require accurate sensors with comparably high update rates, which is not the case with the kind of sensors we typically find in today's ubiquitous computing environments (cf. Table 14.2). Particle filters have been successfully applied to mobile robot localization, and especially the application of real time particle filters [KFM03] appears to be a promising approach with improved resource-efficiency. The complexity of particle filters, however, reduces their suitability for the use with small, resource-limited mobile devices. In our approach, in contrast, we concentrated on providing a lightweight fusion-based positioning architecture with low overhead. As a consequence, the iPOS system was deliberately tailored for the operation on small, resource-limited devices. Furthermore, our fusion algorithm is well suited to work with standard sensors available in typical ubiquitous computing environments and does not require specialized dedicated hardware such as laser range finders or ultrasound badges, for instance. In this context, however, integrating our fusion algorithm with a low overhead version of the Location Stack for explicit support of mobile devices could be an interesting option.

The COMPASS location system [KB05] is a more recent positioning architecture that also takes up the idea of fusing different sensors with the help of a probabilistic fusion algorithm. Similar to our work, they calculate a combined *probability distribution function* (PDF) based on the location information obtained from individual sensors. Instead of using a map model, the PDF is built upon a Cartesian coordinate system. The architecture also features a plugin based design and targets mobile resource-limited devices. The COMPASS location system is further designed to include a so called *translator service* that is capable of translating probability distribution functions or coordinates into meaningful location identifiers such as building names or room numbers. A formal model of the fusion architecture

and the positioning procedure was not presented. Further, so far no prototypical implementation and experimental evaluation of the system is available, but a prototype system that includes two plugins (PS and an accelerator plugin) has been announced as work in progress.

Pfeifer describes a number of fundamental aspects and challenges of redundant positioning in general [Pfe05]. In the process, an abstract comprehensive architecture for redundant positioning is outlined and discussed. However, the approach lacks a formal model of the suggested fusion architecture. Further, no prototypical implementation or experimental evaluation of the architecture is provided.

14.10.5. Centralized Multi-Sensor Positioning Architectures

The iPOS system operates autonomously on a "smart object" and provides applications that are executed locally with location information about that same object. In contrast to this decentralized approach, the *Location Service* [ABO02] is a centralized service providing access to location information about arbitrary entities. It supports a variety of sensors, but doesn't emphasize fusion of data acquired from these sensors. Rather, it seems to consider all sensor input as accurate, though possibly not being complete (e.g., lacking orientation information). By merging different sensor data, it generates complete location updates for tracked objects. The most significant difference to our work is the Location Service's global availability, i.e. it is built to supply all applications with location information, not only the tracked object itself. From a reliability point of view, such a centralized service constitutes a single point of failure, which is not suitable for critical applications.

Leonhardt and Magee suggested a theoretical fusion architecture for multi-sensor location tracking [LM98] based on a formally defined, hierarchical location model. The fusion algorithm can identify overlaps among location sightings and exploit them for increasing the accuracy of location estimations. While the fusion and abstraction of location information is carried out by the clients, the system relies on a central repository for locations and location-objects. An initial prototypical implementation of the location service in the form of a "specialized database" was briefly outlined, which allowed to verify the location model but suffered from some challenging performance and scalability problems.

Baus et al. presented a resource-adaptive mobile navigation system [BKW02]. However, their complex 3D-model-based (indoor) navigation service is not running on the tracked device itself, but resides in the background infrastructure instead. The mobile client is mainly used for user input and as portable display. Also, resource-adaption is limited to presentational aspects with respect to the quality of graphical output to the user.

14.10.6. Multi-Sensor Context-Awareness Architectures

Further related is work on multi-sensor context-awareness architectures, where the main goal is to make mobile devices and smart artifacts aware of their physical, situational, and user-related context in general, or to provide location-aware content. In doing so, the challenge of fusing location information for the positioning of mobile devices is not addressed. Examples for such multi-sensor context-awareness architectures and middleware platforms are [CSG99, PPZ99, GSB02, RAMC+04].

Acknowledgments

The author wishes to acknowledge Harald Vogt for several fruitful discussions on the fusion algorithm and on the modeling of (un-)reliable sensors.

The author also wishes to acknowledge a number of students who contributed considerably to the development and prototypical implementation of the iPOS system and its earlier predecessor system. Most notably are the contributions of Christian Schär [Sch02], who worked on the design and implementation of the original positioning system, René Gallati [Gal04], who implemented large parts of the resource manager and positioning API of the iPOS system, and Marco Feriencik and Matthias Leumann [FL05a], who implemented the revised iPOS fusion engine, completed the porting of the positioning platform to accommodate the particularities of the iPAQ PocketPC platform, and helped with the final experimental evaluation of the iPOS positioning prototype. Further, the author thanks Danat Pomeranets and Stephan Schneider for their work on different sensor plugins [PS03], and Simon Schlachter for his work on movement patterns and position prediction based on the grid map model of the positioning system [Sch03].

15. LuxTraceRT: Real-Time Positioning and Self-Calibration Using Indoor Lights and RFID

No navigation system provides zero error or 100% coverage. By using multiple navigational systems in parallel helps to mask areas where one system fails or, by cross comparison (between a number of systems), increase accuracy and reliability. To investigate this, we combine two novel navigational technologies: radio frequency identification (RFID) and light sensors.

In this chapter, we present our work on LuxTraceRT (short for LuxTrace Real-Time), a context-aware self-calibrating real-time positioning system. LuxTraceRT is a further development of LuxTrace [RAT05], a system that provided first evidence for the feasibility of using solar modules as a low-cost alternative for measuring distance in indoor-settings. The original LuxTrace system uses a light intensity model for the mapping of position displacements to light intensity values. LuxTrace then applies measured light intensities of overhead light sources to its internal model to determine the current displacement offset of a mobile host (i.e., a mobile device or a person) carrying the solar cell. By repeatedly executing this procedure, LuxTrace makes it possible to estimate the position of the mobile host for movements along straight trajectories.

The original LuxTrace [RAT05] system was limited to offline operation and required manual intervention for building the used light intensity models. The LuxTraceRT system, in contrast, has been designed as a self-contained application, capable of calculating position estimates in real-time during runtime. Further, LuxTraceRT is able to learn new models on-the-fly in order to adapt to changes in lighting conditions when moving to a different location. In addition, LuxTraceRT makes use of a super-distributed RFID tag infrastructure as a source of location-dependent context information, which is used for enabling self-calibration of the positioning service, and for improving the quality of learned models. The combination of two positioning techniques (based on solar cells and super-distributed RFID tags) helps to mask areas where one system fails in order to improve reliability and availability, and provides a low-maintenance solution for increasing coverage and energy efficiency.

To evaluate our approach, we performed a number of practical experiments with our prototypical LuxTraceRT implementation. Our results demonstrate that the LuxTraceRT system retains its effectiveness in comparison to the original LuxTrace system even when applied under more practical conditions.

15.1. Motivating Scenario

Owing to advances in micro-engineering, it has become possible to integrate ever more sensor hardware on small embedded computing platforms. Accordingly, the number of diverse physical phenomena of the environment that can be sensed by a single device is increasing. By combining sensors that complement each other in a redundant manner with regard to a certain task, the diversity of sensing capabilities can be exploited by means of multi-sensor data fusion with the result of achieving a fault-tolerant and adaptive system behavior. In the following, we describe a general factory scenario where non-contiguous areas of super-distributed RFID tags are combined with light intensity measurements using a solar cell for providing an adaptive, context-aware positioning system capable of self-calibration in real-time.

> An electric forklift truck is equipped with an on-board navigation computer that guides the forklift driver to quickly find specific locations (i.e., shelves or goods) on the large factory grounds. In order to enable the truck to establish its position within the factory area, two complementary technologies are used: RFID and solar cells. For that, the navigation unit on the forklift truck is connected to an RFID reader and antenna attached to the underside of the vehicle, and to solar cells mounted on its roof. The area in which the truck operates consists of large factory halls with long straight corridors, and passages between these corridors and halls. Each corridor features artificial lighting from evenly spaced light tubes. The floor space of all passages between factory halls or corridors are equipped with a layer of densely distributed RFID tags. Each individual tag knows its position coordinates with regard to the local factory map (i.e., the position coordinates are stored in the physical memory of the respective tags).
>
> If the forklift truck passes over an RFID-tagged area, it determines its position within the factory map by averaging the tag position coordinates of the RFID tags detected underneath the vehicle. Alternatively, based on a previously learned light intensity model, the forklift vehicle uses the solar cell measurements for positioning along straight corridors. A display in the driver's cabin shows a map of the factory area and highlights the current position of the forklift as well as the position of the driver's chosen destination.
>
> As the spacing of the lights in different corridors may vary, different light intensity models are required. Therefore, whenever the forklift truck enters a new corridor, it uses its RFID-based positioning service as a reference for learning a new light intensity model on the fly. It further optimizes learned models by adjusting their period lengths. The period length equals the distance between adjacent light tubes, and this information is also stored on the RFID tags at the entrance to each corridor.
>
> While driving through long corridors towards a certain shelf, the navigation system on the electronic forklift truck turns off the RFID reader and uses the solar cell input for reckoning its positioning. The solar cell

sensor is several orders of magnitudes more energy efficient than the RFID system. This helps the vehicle to extend its battery lifetime, and as a direct result of that also the overall operation time. Whenever the forklift truck gets near a passage to another corridor with potentially different lighting conditions, the RFID system is activated so that the light intensity model can be recalibrated. Thus the use of the existing lighting infrastructure for positioning during straight driving along corridors increases the coverage of the navigation system within the factory area while minimizing the additional hardware costs for the RFID equipment, as only the critical passage areas need to be fitted with RFID tags. Finally, in areas where the availability of positioning information is critical, both the RFID and the solar cell system can be used in parallel to achieve fault tolerance in situations where one system fails.

15.2. Design Goals

The initial LuxTrace development focused on finding a suitable mathematical model for describing the intensity of light tubes at arbitrary points in a room or hallway, and on using that model for deriving displacement and context information [ART05, RAT05]. With regard to positioning, the developed algorithms used in the LuxTrace project are capable of estimating position displacements on straight trajectories using a single solar cell mounted at a constant height, based on a model describing light intensity along a corridor with evenly spaced fluorescent tubes. The model is simplified in so far that it assumes ideal light sources without reflector and diffusor, and without reflections from walls and object surfaces.

A drawback of the initial LuxTrace system was that the light intensity models used for positioning had to be created offline, requiring manual optimizations and post-processing. As a consequence, the calculation of position displacements had to be performed off-line, after the experimental measurements of the solar cell voltage values were completed on the test track.

The primary aim of LuxTraceRT was the development of a more practical prototype system in comparison to the original LuxTrace offline system. The LuxTraceRT system was designed to support real-time operation, enabling the training of new models on the fly for inferring position estimates at run time. A secondary aim was the improvement of the performance of the system by incorporating location-dependent context information.

Using RFID technology contributed to achieving these aims by providing a reference positioning service, and by delivering location-dependent topology information in situ. The latter enabled the self-calibration of the LuxTraceRT positioning service, and the optimization and adaption of learned models.

Overall, the design goals of LuxTraceRT were the following:

- *Real-Time Operation*: Model building and position estimation in real-time based on solar cell voltage measurements using a single solar module as main sensory input.

- *Adaptability*: Adaptive behavior in environments with different lighting conditions by means of continuous *self-calibration* using a third party positioning

service if available.

- *Context/Location Awareness*: Self-calibration by exploiting context information about the physical location. Such information included the metric distance between light sources (for determining the exact length of one period of the light tube intensity model) and parameters for the normalization of light intensities (such as the maximum intensity at a certain reference height).
- *Fault Tolerance* through exploitation of functional redundancy of an available solar cell: in addition to its primary usage of generating energy, the solar cell was employed for positioning. Further, in combination with an RFID-based positioning service, the general availability of position information was improved and the spatial coverage of the combined positioning system increased.

15.3. System Architecture

The LuxTraceRT system consists of three main architectural parts: (1) sensor data acquisition and preprocessing, (2) reference positioning, and (3) real-time model-building and position estimation (Fig. 15.1).

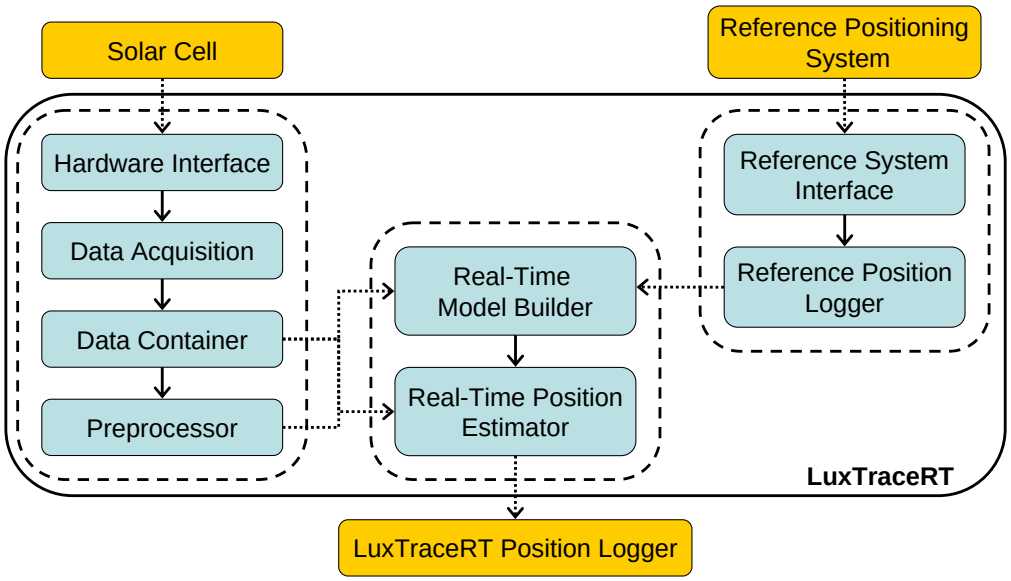

Figure 15.1.: Architecture of the LuxTraceRT system

The *data acquisition* software process continuously samples the digitized voltages from the solar cell module via an A/D converter, which serves as *hardware interface*. The samples are stored in a persistent *data container*, which is monitored by a *preprocessor* software process that detects peaks and lows with regard to the developing of the light intensity voltages.

The main component of the reference positioning part is the *reference position logger*. Through the *reference system interface*, it continuously polls the service of an arbitrary third-party *reference positioning system*, timestamps the received positions, and stores them for later retrieval. The reference position logger further uses linear interpolation to determine intermediate positions for timestamps where no direct position information has been stored.

The core components of the LuxTraceRT architecture are the *real-time model builder*, which is responsible for training new real-time models based on the data made available by the sensor data acquisition and preprocessor units, and the *real-time position estimator*, which calculates position estimates with help of the currently active model. The position estimator loads newly built models during runtime upon availability, thus performing a continuous real-time self-calibration of the LuxTraceRT system.

15.4. System Design Aspects

In the following, we discuss a number of relevant design aspects of the LuxTraceRT system in more detail.

15.4.1. Extracting Context from Indoor Lights

Advantages of extracting context data from indoor lights are that this infrastructure may be used without modification (i.e. the technology is minimally invasive) and this source of context data is generally untapped. Solar cells as sensors for wearable computing are attractive firstly as their low number of "pixels" (e.g., 1) implies that limited processing power may be sufficient. Secondly, as solar cells can be produced in thin (< 1 mm) low weight layers on a flexible substrate in a number of colors they better match garment specifications than other sensing devices. A third benefit of solar cells is that they can be used as a sensor and an energy harvesting device, thus contributing to the restricted energy available on mobile devices and wearable electronics.

15.4.2. Integration of LuxTraceRT with RFID Technology

The decision of employing RFID technology as a complementary technology to the LuxTraceRT system has two major advantages. Firstly, it provides us with a robust and accurate reference positioning service. Secondly, by densely distributing RFID tags in transition areas where the lighting condition changes, which requires that the mathematical light intensity model is adapted accordingly, we can use the memory capacity of these tags for the provisioning of additional location-dependent information in situ. Secondly, the combination of a low-power light sensors with RFID technology, which has power requirements that are several orders of magnitudes higher, demonstrates the possibility of using LuxTrace as a low-power substitute for positioning systems with significantly higher energy demands, and as a low-cost means to improve the coverage and availability in areas where the operation of the primary positioning service is technically or economically infeasible.

15.4.3. Using Context-Information for Self-Calibration and Model Adaptation

Apart from providing location information for the self-positioning of mobile entities, the use of a super-distributed RFID tag infrastructure makes it possible to store arbitrary context information in situ, enabling the realization of *self-describing locations* (see Chapter 11). In our particular case, we are not only able to store local

or global positioning coordinates on the single tags of the RFID infrastructure, but also more detailed information about the topology of location-dependent lighting infrastructures. Such information can include the distance of adjacent light sources, which can be used for the correction of the period length of a learned model. The RFID tags can also provide mobile devices with information on the maximum light intensity per light source at a certain height, which facilitates the adaption and re-calibration of models for the use with solar modules that are placed at different height levels. Ultimately, complete pre-learned models can be stored on the tags of the RFID infrastructure at different physical locations, if the storage capacity of the tags is sufficiently high. This would significantly increase the autonomy of the system by removing the need for connectivity and decreasing the dependence on background service infrastructures.

The RFID tags of the super-distributed RFID tag infrastructure may further be used for providing users with virtual hyperlinks to additional online content residing on a database in the background network infrastructure [WFGH99]. A compromise between retrieving information from a remote database and reading all data from the local physical tag data memory is to provide the mobile device with its own pre-loaded tag database, which can be tailored for the use with certain applications as well as for certain geographic areas and buildings.

15.4.4. Positioning Procedure

LuxTraceRT uses the following procedure for calculating positions: the current position $p(t)$ at time $t > 0$, relative to a given starting point $p(0)$, is determined by multiplying the period length l of the model with the number of complete periods $m(t)$ that so far have been detected while moving from light source to light source along the straight trajectory. Then the displacement offset $o(t)$, which corresponds to the estimated displacement for the current period, is determined by looking up the respective value in the model, based on the most recent light intensity measurement (voltage value) as input (see Fig. 15.2).

The displacement offset is unambiguously defined since the rising and falling flank in the model can be distinguished by considering previously detected highs and lows. The complete equation for the position calculation is given in Eq. 15.1.

$$p(t) := p(0) + m(t) \cdot l + o(t) \tag{15.1}$$

The LuxTrace approach so far only considers straight trajectories for positioning. This implies that the estimated positions are 1-dimensional, measuring the distance along a straight path.

For our experiments, we used a local two-dimensional coordinate system where the x-axis was aligned with the trajectory. Therefore p(t) provided an estimate for the x-coordinate of the current position, while the y-coordinates remained constant. The local position coordinates can be transformed into the format of arbitrary coordinate systems. In our system, we implemented coordinate transformations for obtaining global positioning coordinates according to the WGS-84 standard [Eur06].

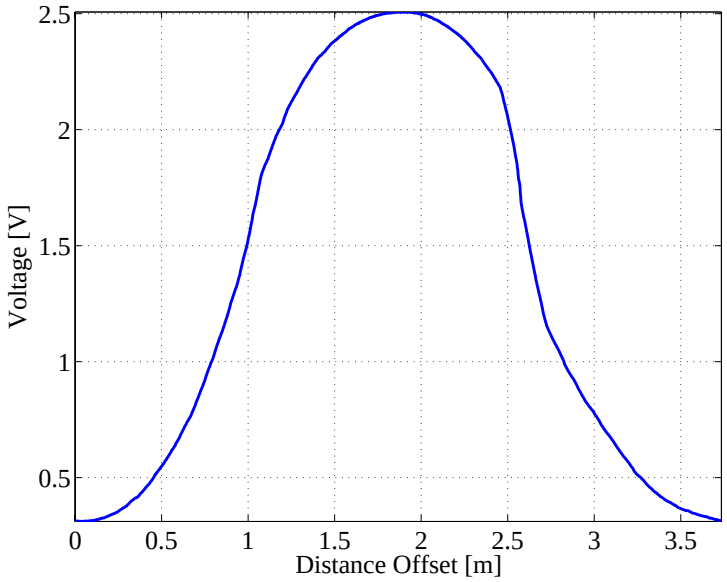

Figure 15.2.: Exemplary LuxTraceRT model used for estimating displacements offsets

15.5. Experimental Setup

The main goals of the experiments were twofold. The first goal was to evaluate the accuracy and robustness of the self-calibrating real-time LuxTraceRT positioning system. The second goal was to analyze the effect of applying context information about the physical location for improving the quality of the models. In our case we used knowledge about the distance between light tubes in order to correct the period of the learned models.

15.5.1. LuxTraceRT Prototype

We developed a fully functional LuxTraceRT prototype based on the architecture described in Section 15.3.

The sensor hardware and the A/D converter used in the experiments are the same as the ones used during the earlier LuxTrace experiments [RAT05].

We have used the RFID positioning system described in Chapter 13.4.3 as a reference positioning service. It consists of two major physical components: Firstly, a trolley equipped with an off-the-shelf RFID tag reader and with an RFID antenna, which is mounted at the bottom of the vehicle. Secondly, a prototypical super-distributed RFID tag infrastructure, which provides a dense distribution of RFID tags across certain areas of the floor space. The basic operating principle of the RFID positioning service is as follows: it first reads the tag position coordinates stored on the single RFID tags within antenna range. Then the position (x_e, y_e) is computed as the arithmetical mean of the obtained single positions as denoted in Eq. 15.2, where (x_i, y_i) are the position coordinates of n RFID tags detected during one scan pass, with $i = 1 \ldots n$.

$$(x_e, y_e) := (\frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n}) \tag{15.2}$$

The RFID hardware consisted of a mid range RFID reader[1], and an external mid range RFID antenna[2]. As transponders, we employed Philips I·CODE [Phi06] high-frequency (HF) tags[3], with a dimension of 7.5 × 4.5 × 0.1 cm.

The LuxTraceRT equipment was added to the RFID positioning trolley (Fig 15.3). The RFID antenna, not visible in the figure, was attached underneath the bottom pane, at 10 cm above the floor space. At this distance, the approximately square operating area of the RFID antenna was about 50 × 50 cm.

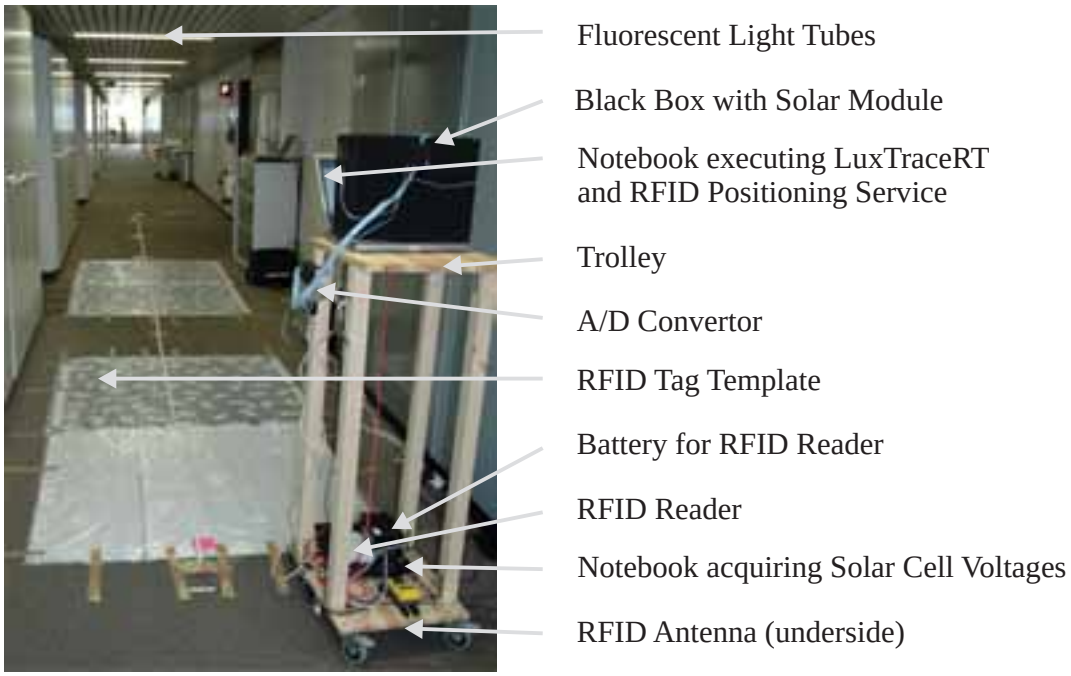

Figure 15.3.: Measurement trolley and prepared test track

For our experimental setup, we used the super-distributed RFID tag infrastructure prototype described in Chapter 13.3.1, which consisted of four RFID-tagged templates with an average tag density of 39 tags/m^2. An overview of the properties of the used RFID templates is given in Table 13.1.

For our practical experiments we chose an office corridor of similar type as the one used for the LuxTrace series of experiments, with light tubes of identical type, arrangement, and light emitting characteristics. To establish our prototypical super distributed RFID tag infrastructure, we installed the four available RFID-tagged foil templates along the length of the corridor. Then we defined the test track by laying out a measuring tape along a distance of 10 m, starting at the beginning of the first RFID template (Fig. 15.3). This tape passed directly underneath the light tube centers. The exact layout of our experimental setup is shown in Fig. 15.4.

The photovoltaic solar module[4] used for the experiments is the same as used for the original LuxTrace system, and it consists of an amorphous silicon thin film deposited on glass. The photovoltaic solar cell was mounted on the trolley at a height of 144.5 cm above the floor, and at 93 cm from the ceiling of the office corridor.

[1]Manufacturer: Feig Electronic, model: OBID *i-scan* reader HF ISC.MR100
[2]Manufacturer: Feig Electronic, model: OBID *i-mid* antenna ISC.ANT340/240
[3]Manufacturer: Philips Semiconductors, model: I·CODE (Type 1)
[4]Manufacturer: RWE SCHOTT Solar, model: ASI 3 Oi 04/057/050

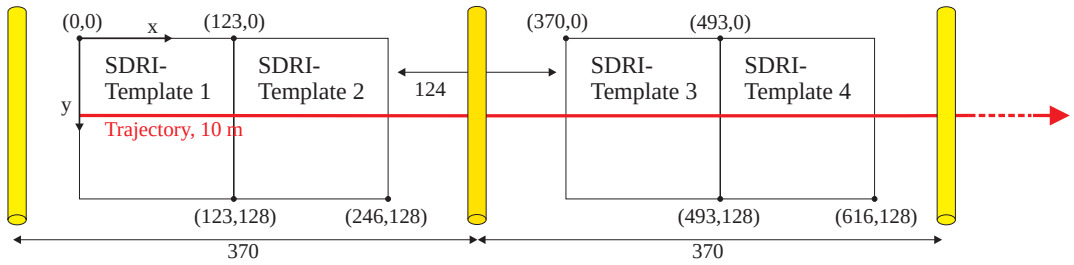

Figure 15.4.: Experimental setup: placement of light tubes and RFID-tagged foil templates (all coordinates and distances in cm). The test track starting at position (0,60) is indicated as a red arrow

15.5.2. Experimental Method

We limited the observation range during the model learning procedure to a single low-high-low cycle of light intensities, as it is desirable to minimize the length of the corridor needed for the training of new models.

The placement of the RFID templates was chosen in a way that allowed us to obtain RFID position information in the critical sections halfway between adjacent light tubes, which is necessary to enable an accurate detection of low points in the measured light intensity.

All practical experiments were performed by pushing the measuring trolley at a constant walking speed of approx. 0.34 m/s (60 steps per minute, pace set by means of a metronome) along the test track.

In order to assess the positioning errors of the LuxTraceRT system under realistic conditions, each trained model would ideally be validated by comparing its position estimates to the actual trajectory along the test track a number of times, exposing it to a different light intensity trace during each pass.

For practical reasons, we experimentally measured and recorded multiple light intensity data streams along the test track. We then replayed these recorded experimental data streams in a simulation mode and compared them with the different recorded reference positioning data streams. We used this feature to simulate further experiments and thus obtain additional results, for example by cross-combining data streams from different experiments. We also employed the simulation mode for analyzing the effect of correcting the period lengths of the real-time models during the model-building process, and for optimizing the peak-detection algorithm.

15.6. Experimental Results

15.6.1. Evaluation of RFID Positioning System

We have evaluated 5 experiments with the RFID positioning prototype, which was mounted onto the trolley together with the LuxTraceRT system. On average, 67 position estimates were obtained per test run along the 10 m test track of which 4.92 m were covered with RFID tags.

The mean absolute RFID positioning error for the five experiments was 10.2 cm, with a standard deviation of 4.3 cm.

Model	period dev.	abs.per. dev.	max.pos. POEE	max.neg. POEE	mean POEE	max.abs. POEE	mean abs. POEE
1	3	3	29	−16	−2	29	9
2	−13	13	27	−29	−8	29	12
3	−2	2	37	−21	−3	37	12
4	0	0	33	−20	−1	33	12
5	9	9	34	−17	2	34	12
average	−0.6	5.4	32.0	−20.6	−2.4	32.4	11.4
std. dev.	8.1	5.4	4.0	5.1	3.6	3.4	1.3

Table 15.1.: Period and position offset aberrations of five real-time generated LuxTraceRT models with variable period lengths, compared to the original LuxTrace model: effective and absolute deviation of period lengths from true period length; positive and negative maximum, and mean POEE; maximum and mean absolute POEE. (All values in cm)

The absolute position estimation error (APEE) over the five experiments was below 16 cm in 70%, below 18 cm in 80%, below 21 cm in 90%, and below 23 cm in 95% of all RFID position measurements. The overall maximum APEE was 26.3 cm.

For comparison, the reference positioning service used for building the original LuxTrace model featured a cumulative position estimation error of below ±2% compared to the traveled distance. With regard to our test track of 10 m length, the maximum cumulative LuxTrace reference positioning error is in the range of ±20 cm, and for a single model period of 3.70 m in the range of ±7.4 cm.

15.6.2. Comparison of LuxTraceRT Models with Original LuxTrace Model

The original LuxTrace model was the result of merging the data from 10 experimental measurements of two subsequent light intensity periods, which were stretched to match the correct distance of 3.70 m between adjacent light tubes.

As the LuxTraceRT system only uses the solar cell voltage data obtained from a single period to train a new model, we compared the position offset estimation error (POEE) of the experimentally generated real-time models relative to the offset obtained from the LuxTrace offline model for given light intensity values.

We performed five practical real-time model training experiments with the LuxTraceRT system on our prepared test track. We then analyzed the aberrations of the period lengths of the models, and the relative errors of the position offsets in comparison to the original LuxTrace model.

An overview of the resulting relative position offset estimation errors (POEE) is shown in Table 15.1. The measured absolute POEE amounted to 11.6 cm with little variation over all five evaluated models, considering a comparably low standard deviation of 1.3 cm, with an average maximum deviation of over 32 cm. Figure 15.5 shows an example LuxTraceRT model with visible deviation from the original LuxTrace model.

The data of Table 15.1 also shows that the models built by the real-time model builder differ significantly with respect to their period length: the absolute period

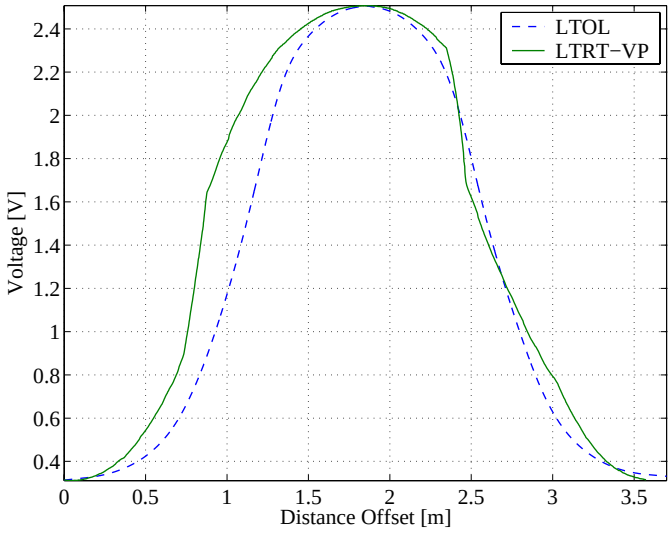

Figure 15.5.: Exemplary LuxTraceRT model compared with the original LuxTrace model

deviation from the true period length was approx. 5 cm, with a standard deviation of approx. 5 cm. In our case this meant that *without* period correction, the LuxTraceRT positioning procedure as defined in Section 15.4.4 suffered from a *cumulative error* of 5 cm on average (based on the five experimental real-time models), and of 13 cm in the worst case (based on the particular LuxTraceRT model depicted in Fig. 15.5), per each 3.70 m of traveled distance.

15.6.3. Effect of Period Correction on LuxTraceRT Positioning Accuracy

We observed a constant distance between the adjacent light tubes in the corridor in which we made our experiments as well as between the light tubes in the corridors of other buildings at ETH Zurich. In case this distance is known, any model created in such an environment can be transformed into a corrected model that has the correct period length. We refer to such corrected models as *fixed period* (FP) models. To evaluate the performance of corrected FP models, we firstly executed the real-time position estimator separately with each of the five experimental LuxTraceRT models with *variable period* (VP) lengths. We performed four experiments with each model in simulation mode, by feeding in four different light intensity data streams (voltage values), which had been recorded with the measuring trolley along the test track.

In a second step, we corrected the period width of the five LuxTraceRT models, by stretching the models according to the displacement of the actual period length from the true period length (which was exactly 3.70 m for our office corridor). The resulting five LuxTraceRT models with fixed period lengths were again evaluated using the four light intensity data streams.

For the analysis, we calculated the absolute and mean position estimation errors (PEE) for the position estimates that we obtained from the different simulated executions of the real-time position estimator. A summary of the results is presented in Table 15.2.

Variable Period	max. pos. PEE	max. neg. PEE	mean PEE	max. abs. PEE	mean abs. PEE
average	26.2	−33.8	−9.0	42.8	17.5
std.dev.	19.8	12.1	13.6	10.7	6.0
Fixed Period	**max. pos. PEE**	**max. neg. PEE**	**mean PEE**	**max. abs. PEE**	**mean abs. PEE**
average	24.5	−31.6	−9.5	33.5	13.9
std.dev.	6.8	7.9	7.0	6.2	4.2

Table 15.2.: Averages and standard deviations for maximum positive, negative, and mean PEE values, and for maximum and mean absolute PEE values of simulated positioning experiments with five LuxTraceRT models, each with variable and fixed period length, using four recorded solar cell voltage data streams as input

Applying the period correction did not significantly improve the mean position estimation error (PEE) itself, which even slightly increased from 9.0 cm to 9.5 cm, but significantly reduced its standard deviation by nearly 50% from 13.6 cm to 7.0 cm. The same observation holds true for the average maximum positive and average maximum negative errors. This stabilization effect was caused by a general harmonization of the maximum and mean absolute errors as a result of the period correction. This manifested itself in a noticeable reduction and stabilization of the maximum and mean absolute PEE: the maximum absolute PEE sank from 42.8 cm to 33.5 cm, with a significantly lower standard deviation of 6.2 cm compared to 10.7 cm. The mean absolute PEE diminished from an average of 17.5 cm to 13.9 cm, together with a 30% decrease of the standard deviation.

We conclude that period length correction significantly improved the accuracy of the positioning procedure. It further resulted in a neutralization of the cumulative error component e in the positioning procedure, which was caused by an incorrect period length $l' := l + e$ in Eq. 15.1 in Section 15.4.4, with l being the true period length.

15.6.4. Comparison of Positioning Accuracy Between LuxTraceRT and LuxTrace

We compared our results for LuxTraceRT mentioned above with those of the original LuxTrace system.

Figure 15.6 shows the measured cumulative absolute positioning errors for the LuxTraceRT system with variable and fixed period lengths, as well as for the original LuxTrace system.

Based on this data we can make the following observations: The LuxTraceRT system with VP models (LTRT-VP) has a significantly lower performance in comparison compared to the LuxTraceRT system using FP models (LTRT-FP) and the original LuxTrace system (LTOL). Nevertheless, for the quantiles $\leq$80%, the maximum error of the LTRT-VP system does not exceed the maximum errors of the other two systems by more than 8 cm.

Comparing the LTRT-FP system with the LTOL system, we can see that the

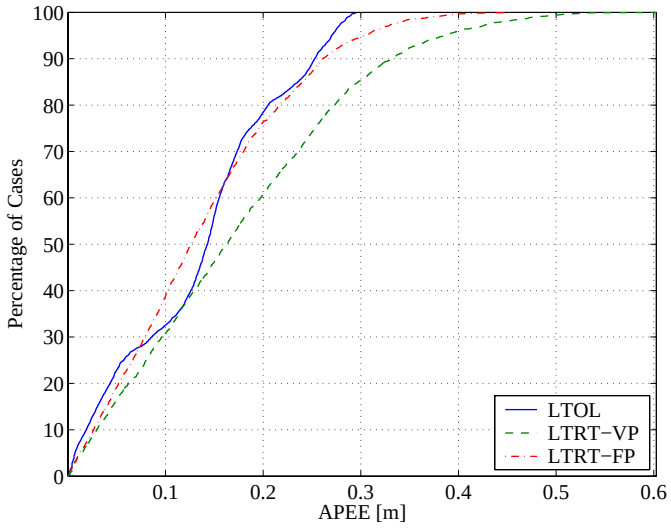

Figure 15.6.: Cumulative absolute positioning errors of LuxTraceRT using VP models (LTRT-VP) and FP models (LTRT-FP), and of the original LuxTrace offline system (LTOL)

maximum absolute positioning error of the LuxTraceRT system employing period length correction is only marginally bigger than the error of the original LuxTrace system (LTOL) for the 60%–90% quantiles. For the 80% and 90% quantiles, for instance, the accuracy of position estimates remain below 22 cm and 26 cm for the LTRT-FP system, and below 21 cm and 25 cm for the LTOL system. We consider these results an indication for the effectiveness and robustness of the adaptive real-time system.

15.7. Discussion

We assess the results of the practical experiments with respect to the location technology assessment taxonomy proposed by Hightower in the Location Stack project [HB01] that considers scalability, cost, recognition, and limitations.

Scalability: A scalability factor related to cost is that as the solar cell devices and the passive RFID tags are relatively cheap, it can be anticipated that the number of these devices, both solar cells on the body and RFID tags in the environment, should not be a limiting factor. For the purely solar cell based LuxTrace system, a wide application range of office and manufacturing hall environments can be expected. The combined LuxTraceRT system improves on the scalability of LuxTrace, as full coverage with both the solar cell and RFID systems is not required. Also, as RFID tags are fixed, they help to reduce the risk of drift from the solar cells.

Cost: For the LuxTrace system consisting of a solar cell and acquisition unit, a low cost implementation can be achieved. The global cost of the LuxTraceRT system includes incremental RFID hardware, installation, and maintenance. For a wearable device, for instance, we would anticipate that one or more antennas built into the shoe(s) would be used. This receiving device would be

the highest power (Watt) incremental sensing device, and overall design would seek to minimize its duty cycle by switching when possible to the solar cell system (Milliwatt) only. The corollary of such reduced power consumption is reduced mass, volume and cost for the battery; such components often have the highest values of the latter criteria in mobile systems.

A scenario minimizing the installation cost of the RFID tags is achieved, when they would be delivered as part of the carpet in a new building. Such tags could also be retrofitted under the existing carpet by using foils as they were used in the experiments of this work. A further way to minimize cost would be to install the tags only at intersections (e.g., corridor-corridor or corridor-room).

Further hardware costs are avoided as indoor lights and on-body computers (e.g., mobile phone) are generally available; incremental maintenance costs of the system would be minimal assuming that lighting infrastructure is not significantly changed and bulb replacement service(s) exist.

Recognition: We have shown that recognition has not suffered despite the more practical scenario of the online model training and position estimation applied in the LuxTraceRT system. Furthermore, combining the solar cell and RFID system improves reliability by supporting calibration of one system against the other. In this way our work presents a simple but robust feature fusion approach of the two systems. We manage the navigation system energy consumption and accuracy as follows: For maximum accuracy we can use both systems whilst for energy saving or for wearable computing scenarios, we selectively use only the lowest power (solar cell based estimator) system.

Limitations: The reported experiments focused exclusively on indoor applications in the absence of natural light. While this is sufficient for many room and corridor scenarios further work is necessary to determine the restrictions incurred from natural light sources.

15.8. Conclusion

We presented a concept to support dependable location tracking for indoor wearable or mobile applications using solar cells and a super-distributed RFID tag infrastructure. We developed a combined system consisting of a displacement estimator based on light sensing and the information collection from carpet-like RFID tags distributed in the environment.

The displacement estimator learns new lighting schemes online if RFID context information is available. We found that the accuracy of the combined LuxTraceRT system was similar to the accuracy of the original LuxTrace system. Using real-time trained models with variable period lengths, the positioning accuracy achieved by the LuxTraceRT system was better than 28 cm in 80% of all calculated estimates on straight trajectories. By incorporating context information (distance between adjacent tubes) from the RFID infrastructure, this accuracy improved to less than 22 cm in 80% of all measurements. This compares well to the accuracy of less than 21 cm for the original LuxTrace system under comparable conditions (80% quantile).

We conclude that for the controlled conditions in which we experimented, the use of solar cells alone is generally sufficient. An example application for the offline LuxTrace system would be a linear displacement measurement on a track. However, for location tracking applications with less prior knowledge (e.g., spacing of lights, starting position, less predictable trajectories) the combined LuxTraceRT system may prove more accurate and practical.

With regard to dependability, the LuxTraceRT system provides an example for exploiting functional redundancy: in the LuxTraceRT system, solar cells are not only employed for energy generation, but also for the generation of location information derived from light intensity measurements, which then was used for calculating position estimations. Further, the LuxTraceRT system serves as an example of a hybrid system that integrates the two concepts of redundant-sensor data fusion and localized cooperation for achieving fault-tolerant behavior. Firstly, by combining the solar-cell based positioning system with RFID positioning capabilities, we obtain a redundant information fusion system. Secondly, by making use of location-dependent context information obtained from a super-distributed RFID tag infrastructure, we could significantly improve the quality of service of the model-building and of the position estimation process.

15.9. Related Work

The limitations of existing technologies are related to a number of factors including coverage, cost, and location estimation accuracy. Satellite navigation for indoor application is unsatisfactory in all three categories given that it is generally not available indoors, is relatively expensive, and the error of the Global Positioning System (GPS), for example, is of the order of 15-20 m [Get93]. This error can be reduced to 1-2 m by error correction using the European Geostationary Navigation Overlay Service [Bre03]. The European Galileo [EC05] system, planned to be commercially available by 2008, targets a global accuracy on the horizontal plane of up to 4 m for safety of life services and mass market applications, and up to 1 m for commercial applications. Galileo may offer horizontal accuracy in the centimeter-range by means of locally augmented signals [EC03]. Numerous other location tracking technologies exist that rely on absolute position estimations, such as radio frequency identification (RFID) tags [NLLP04] or wireless communication systems [VWG+03, AKS04]. Generally these technologies rely on a supportive infrastructure. Relative position information include ultrasound [AAA97], and inertial sensors [VMKA02]. These systems suffer from continuous drift since no re-calibration mechanism is integrated.

Acknowledgments

The author wishes to acknowledge Martin Burri for his work on the implementation and experimental evaluation of the LuxTraceRT prototype system [Bur05]. The author further wishes to thank Oliver Amft and Julian Randell for fruitful discussions on design aspects of the original LuxTrace system.

Part IV.

Social Perspective on Dependability

16. The Social Dimension of Dependability in Ubiquitous Computing

Life without computers is unimaginable for most of us today – embedded processors monitor the condition of high-risk patients around the clock, they control central heating in buildings, air conditioning in tunnels, and they safely guide airplanes between continents. The potential economic benefits of ubiquitous computing are certainly key factors for the further proliferation of information technology, such as novel indoor and outdoor positioning systems, ubiquitous communication platforms, and unobtrusive monitoring installations. This technology will form and shape the foundations of future ubiquitous computing landscapes.

As more and more objects and environments are being equipped with ubiquitous computing technology, the degree of our *dependence* on the correct, reliable functioning of the deployed devices and microcomputers including their software infrastructures is increasing accordingly. Today, in most cases, we are still able to decide for ourselves whether we want to use devices equipped with modern computer technology (e.g., by choosing manual control for our central heating, or by deciding not to carry a mobile phone if we dislike the constant accessibility its usage implies). But in a largely computerized future, it might not be possible to escape from this sort of technologically induced dependence. This leads to a number of fundamental social challenges for future ubiquitous computing systems. Privacy [Lan05], for example, is a very prominent challenge. However, the more thoroughly "computerized" our environment becomes, the more basic attributes of the world we live in will subtly change, such as its reliability, accessibility, and transparency. In the following, we attempt to identify these dependability-oriented concerns and try to address ethical and social implications of future ubiquitous computing landscapes.

16.1. Reliability

As we have seen earlier in Chapter 3, the vision of ubiquitous computing describes systems that aim at working in the background, discreetly and unobtrusively helping us to carry out our tasks. Since our needs and circumstances can change over time, such systems must be able to adapt themselves dynamically to the current situation (see Sect. 4.6). In doing so, one crucial basic requirement is *reliability in the broadest sense* of the word. In addition to ensuring dependability from a technological point of view as described in Chapter 4, a complex and highly dynamic system must also remain manageable and controllable, and must retain the ability to predict (and, to a certain extent, verify) that the system is behaving correctly.

16.1.1. Manageability

It is far from clear that implementing large-scale ubiquitous computing scenarios involving potentially millions of smart, adaptive devices, is simply a question of scaling up existing toy examples. As the number of interacting objects increases massively, it has to be ensured that the services and applications based on these objects still are able to meet their original requirements. And, above all, it is important that it remains possible for people to understand and control a highly dynamic ubiquitous computing world involving such large numbers of individual objects and devices.

16.1.2. Predictability

Today's technical infrastructures, such as the phone system, television, and electricity, are relatively easy to use, even for people with no special qualifications. This also entails the ability to detect malfunctions: for example, if you lift a telephone receiver and do not hear a dial tone, it is immediately evident that the phone (either the handset or the landline) is not working properly. However, this type of predictability of system behavior can no longer be taken for granted in a ubiquitous computing landscape, as systems are expected to function without users noticing their presence. This will make fault detection and diagnosis fundamentally difficult, especially for the layman [ECPS02]. Additionally, users might continue to rely on a failed service (e.g., an automated backup service or the self-diagnostics of a smart product) without noticing, thus possibly increasing the damage done until the problem is finally discovered.

16.1.3. Dependability

Incorporating computing and communication technology into everyday artifacts requires ever-decreasing form factors and minimal energy consumption. This makes it difficult to use hardware redundancy in such systems, even though the envisioned unobtrusive and ubiquitous use of these systems implies much harsher surroundings than, say, an everyday indoor environment. We have already seen some alternative concepts and mechanisms earlier which help to overcome service interruptions and device failures, such as *localized resource sharing* (Sect. 9.2), *fault-tolerant data fusion* techniques (Sect. 9.3), *instant personalization of mobile devices* (Sect 6.4), and the explicit *diversification of system functions*, which can provide fully independent ways of carrying out the same task, preferably based on separate sets of system resources wherever feasible (Sect. 6.3).

16.1.4. Conclusion

The power outages that affected not only large parts of the USA and Canada but also Italy and some other countries in 2003 have demonstrated our dependence on existing technical infrastructures, in this case the power grid. With the constant goal of saving costs, any industry-built ubiquitous computing infrastructure is liable to run a high risk of forgoing safety for the sake of efficiency, possibly resulting in brittle systems that will work only sporadically. Ensuring the reliability of such systems in the broadest sense therefore is a crucial challenge.

16.2. Delegation of Control

In order to minimize the need for human intervention in complex, highly dynamic environments, new concepts for *delegating control* are necessary – automatic processes should take care of routine tasks in a dependable manner, but also provide accounting mechanisms for monitoring the increasingly complex control flows. Control and accounting mechanisms are important tools for determining who is in control of an autonomous system, and who is responsible if something goes wrong. At the same time, however, the autonomy of artifacts is also limited by their reliance on the technical infrastructure.

16.2.1. Content Control

If smart objects provide information about themselves, this raises the question of who guarantees the objectivity and accuracy of the statements made. For example, smart products might be used to tie customers more closely to traders by making these products recommend the purchase of other goods produced by the same trader. In a certain sense, smart objects are becoming media representing a particular "ideology" (e.g., that of the product's manufacturer, or the politically motivated opinion of a consumer protection organization). For instance, a consumer protection institute could use its own electronic directory to map a smart product label onto information other than that which the producer intended (for example, to warn of allergies to ingredients). And maybe more importantly, this raises further questions: for instance, who will decide what a smart toy tells the children, potentially shaping the children's opinions without their parents' knowledge. Tempting children to buy additional toys would only be the most obvious strategy – a much more serious threat could be the moral values induced by smart toys during play.

16.2.2. System Control

It is similarly conceivable that automobiles or other products, as components of an ambient intelligence network, would no longer feel completely "loyal" to their owners, but would instead enforce the guidelines of insurance companies, manufacturers, or the judiciary. For example, a smart car might refuse to open the door for its driver because he or she has stopped in a no-parking zone. One fundamental question is about when an intelligent device should obey human orders, and when it should follow its own "convictions". While such a no-parking system might be desirable for congested cities, some kind of manual override mechanism would obviously be needed for emergency situations, e.g., when rushing a seriously injured person to hospital (and trying to park in front of it). Even if a system were designed to only make suggestions, it would still find itself treading a fine line between inspiration and frustration, between obliging helpfulness and pigheaded patronization [Sat01].

16.2.3. Accountability

If autonomous objects such as the previously mentioned smart doll start taking decisions on their own (e.g., buying new clothes), legal guidelines need to be drawn

up in order to resolve who is ultimately responsible for these business transactions. Smart assistants might order unwanted plane tickets, smart fridges excessive amounts of food – in both cases the automated system might have performed according to specification, though neither the original programmers nor the layman user would be able to understand its reasoning. Providing the user with a detailed explanation of completed transactions is only part of the solution, especially when monetary damages are involved. It may look appealing to simply shift the responsibility and liability onto the end user by changing the license agreements of smart objects accordingly in the small print, but it is questionable whether such a procedure would prove tenable if taken to court.

16.2.4. Conclusion

Similar discussions to the ones described above, involving the questions of accountability and content control, are already taking place in the context of the World Wide Web. For example, questions regarding the right to possess and use certain prestigious domain names [DNH04, Bit05] can be compared to the issue of content control (i.e., who is allowed to resolve a certain URL stored in the RFID tag of a product), while national laws trying to control digital copyright as well as freedom of speech [Ele06] might already set standards regarding the future "freedom" of smart devices to obey their owners.

16.3. Social Compatibility

Another fundamental challenge for ubiquitous computing systems is their *social compatibility*. If we, as humans, want to be capable of participating in highly dynamic systems, the parameters of these systems will have to be adjusted accordingly. System behavior relating to particular aspects should retain a certain transparency and inertia, allowing humans to detect and adjust to changes. On the other hand, it should be taken into account that an all-encompassing ubiquitous computing landscape must also meet the needs and requirements of as broad a section of society as possible, especially if participation in such a computerized landscape is practically mandatory.

16.3.1. Transparency

In a future full of smart objects, we may not only have the *ability* to shop just about anywhere at any time, we may also be *compelled* to buy just about everywhere, all the time: ubiquitous computing makes it possible to implement novel *pay-per-use models* based on everyday objects equipped with sensors and communications capabilities. Furniture, for instance, could monitor its usage (e.g., a sofa could count the number of persons that sit on it, the persons' weight and seating time) and create a monthly itemized billing statement. The ubiquitous application of pay-per-use schemes, however, might incur a large number of micropayments, e.g., for bus or theater seats we have sat on, pages of books or newspapers we have read, or clothing we have worn. Irrespective of technical feasibility, this prompts the question of how we could keep track of the resulting number of short-term contracts and the countless associated micropayments, let alone retrospectively

check the legitimacy of these transactions. Not only would it be extremely tedious and unrealistic to manually check thousands of transactions, it is also questionable to what extent inappropriate items could be identified and rejected, and to what extent legitimate payments could be unambiguously and indisputably allocated to the responsible party. Dynamic insurance rates that may vary according to the style of driving [CH04, Cor06] constitute another example of a potential loss of *transparency*, especially if the underlying assessment methods changed dynamically without warning, or if they were unknown or too complicated to be understood by the user.

16.3.2. Knowledge Sustainability

Most information in our everyday life today remains valid for an extended period of time, such as food prices in our favorite supermarket, or prices for public transport, for example. It is this inertia of information that permits us to use acquired knowledge and prior experiences to cope with future situations and tasks. In a highly dynamic world, the sustainability of knowledge risks being lost – an experience that was valid and useful one minute could become obsolete and unusable the next. Such a loss or accelerated devaluation of long-term experiences could, in the long term, contribute to an increased uncertainty and lack of direction for people in society.

16.3.3. Fairness

Detailed cross-marketing based on ubiquitous computing promises tailor-made offers that virtually eliminate unwanted advertisements. However, a specific offer may be withheld from a particular consumer for one of two reasons: either the offer was not worth the consumer, or the consumer was not worth the offer. David Lyon, Professor of Sociology at Queen's University in Canada, calls this process "social sorting" – "Categorizing persons and groups in ways that appear to be accurate and scientific, but which in many ways accentuate differences and reinforce existing inequalities" [Lyo01]. People not matching a certain 'desirable' profile might have to pay much higher prices, as they do not qualify for any of the existing discounts, which might in turn reinforce the non-matching patterns.

16.3.4. Universal Access vs. Digital Divide

The natural interfaces envisioned in ubiquitous computing scenarios certainly have the opportunity to overcome many of today's accessibility problems, such as the issue of small screens and keypads of modern mobile phones that often prevent elderly people from using them [Gon01]. Today many projects in the field target elderly and physically disabled people in particular, providing them with electronic "memory aids", reading aids, orientation and navigation systems [Mak01, CKR04], for instance. These projects might pave the way for a universal design [Ste01] that considers the needs of minorities and marginal groups early in the design stage. Intelligent interfaces and the concept of ubiquitous information access are often seen as key developments for bridging the digital divide, where different sections of the population have different abilities to participate in the information society.

However, having more information opportunities does not necessarily mean more justice or freedom, simply because the potential dependencies and opportunities for manipulation would be so numerous they could overwhelm individuals, making it even more difficult to assess the trustworthiness of the source of the information. Information that was uncritical or sponsored by advertisers (and therefore one-sided) could become available free of charge, while independent, high-quality information would cost money, thus widening the digital divide even further. Since ubiquitous computing is not just about information itself, but is inherently linked to real-world objects, these new means of access and content control could easily lead to the digital divide becoming a real and perceivable rift in our everyday lives.

16.3.5. Conclusion

In history, the development of regulatory, social, and ethical standards tends to lag considerably behind the rapid proliferation of pioneering technological inventions, as was the case with the invention of the assembly line and mass production at the beginning of the 20th century, and with the appearance of the global Internet in the 1980s, for example. In an emerging future of ubiquitous computing systems, one exciting question is whether we will be aware of the impending pitfalls and tackle them in an early (design) phase, where we still have the means to shape the envisaged systems according to fundamental social and ethical requirements, or if there is a need for yet another social revolution that subsequently brings about necessary adaptations by force.

16.4. Acceptance

The fundamental paradigm of ubiquitous computing, namely that computers disappear from the user's consciousness and recede into the background, is sometimes seen as an attempt to have technology infiltrate everyday life unnoticed by the general public in order to circumvent any possible social resistance [Ara95]. Yet beyond any perceived sinister motives (which might be easy enough to counter), a widespread public *acceptance* of ubiquitous computing also rests on issues of an almost philosophical nature, such as the fundamental nature of smart objects or our changing relationship with our environment.

16.4.1. Feasibility and Credibility

Many philosophers and social scientists identify a prevailing self-confident and technophile attitude among scientists in the field of ubiquitous computing [Ada00], where the non-critical anticipation of future technological developments almost attains the characteristics of a metaphysical prophecy. Others doubt the credibility of the envisioned scenarios, e.g., when ambient intelligence is said to simplify our lives, help us save time, and relieve us of laborious tasks. While this assertion has been constantly repeated throughout the twentieth century by the consumer goods industry, adding "smart machines" everywhere will not help to overcome the existing pattern of hurry, rush, stress, and separation from other people, but will only increase their efficiency [Win99]. Such criticism may build up and induce a serious credibility gap, reducing the acceptability of ubiquitous computing technologies.

16.4.2. Artifact Autonomy

Networked everyday objects embedded in a ubiquitous computing landscape lose part of their autonomy and, with this, exhibit an increased dependence on the infrastructure. For users, this reduces the "object constancy" of the objects that surround them, as the example of electronic books made from smart paper shows: reading such a book may presuppose a regular connection to a server (license server, user account server, etc.). Because of this, an e-book appears to be more error-prone and less autonomous than a "normal" book, which can always be read, whereas the electronic one can only be read if the required background infrastructure is functioning and reachable.

16.4.3. Impact on Health and Environment

It is hard to predict the impact that a large-scale use of ubiquitous computing technology would have on our environment in terms of raw material consumption, energy consumption, and disposal. For example, if all supermarket goods were equipped with smart labels in the future, billions of these tiny and individually quite harmless chips would end up in the household garbage. On the other hand, the remote identification capability provided by smart labels would enable information on products to be made available throughout their entire life cycles, permitting the different materials in waste products to be efficiently identified and separated. It is also not yet fully understood whether, and to what extent, electromagnetic radiation (e.g., produced by wirelessly communicating smart objects) could affect our physical health. A vision involving myriads of everyday objects and wearable "information appliances" that communicate wirelessly with each other thus gives due cause for concern, as its potential adverse environmental effects could permanently influence the lives of future generations [LSM+03].

16.4.4. The Relationship between Man and the World

From a philosophical point of view, the vision of ubiquitous computing fundamentally changes the environment in which we live: "By this weaving of extensions of ourselves into the surroundings, significant parts of the environment lose important aspects of their otherness and the environment as a whole tends to become more and more a subservient 'artifact'. This artifact, which the world immediately surrounding us becomes, is almost entirely 'us' rather than 'other'. In this sense, the surrounding world has almost disappeared." [Ara95]. Similarly, Adamowsky stipulates that our inability to handle the physical world in a flexible enough way will force us to replace it by digital surrogates – equivalents of particular aspects of the real world in the digital world, implemented in the form of models, simulations, and virtual counterparts – which will ultimately lead to a transformation, dislocation, substitution, and the loss of fundamental properties relating to the world [Ada00].

16.4.5. Conclusion

Dryer et al. [CCS99] conducted two empirical studies to examine the theoretical relationships between system design for mobile computing, human behaviors, social attributions, and interaction outcome. In their conclusion, they express "doubt that

our inevitable future is to become a machinelike collective society. How devices are used is not determined by their creators alone. Individuals influence how devices are used, and humans can be tenaciously social creatures." They conclude "Given the importance of social relationships in our lives, we may adopt only those devices that support, rather than inhibit, such relationships." With the substantial amount of skepticism related to technology, such findings seem to counterbalance the immediate threat that a thoroughly computerized future appears to hold. However, apart from personal prejudices, the wide range of social consequences that ubiquitous computing may have will certainly need to be addressed in future systems and debates. These challenges are of fundamental importance and may ultimately even have a decisive influence on the large-scale acceptance of ubiquitous computing technologies and environments.

16.5. Living in a World of Smart Environments and Augmented Objects

The augmentation of everyday objects and physical spaces with sensing, computing, and communication capabilities per se has a priori no negative implications. In the following, we focus on the augmentation aspect of ubiquitous computing, pointing out benefits, problems, and possible technical solutions.

16.5.1. Benefits of Smart Spaces and Augmented Objects

As we have seen in the previous chapters, augmented real world objects and smart spaces can contribute to the realization of novel services and applications that are to the benefit of the individual and the society as a whole. An individual person, for instance, may consider the provisioning of context-dependent information while moving through smart environments (e.g., tour guides, navigation systems, or virtual collaborative work spaces) convenient and helpful. Furthermore, everyday environments augmented by ubiquitous computing technology can also provide means to alleviate the difficulties and disadvantages of marginal groups who find themselves at the fringe of society. Projects targeting elderly and physically disabled people enable such persons, who are often neglected as marginal groups, to participate more actively and autonomously in everyday life. A concrete example is the Chatty Environment [CR03, CR04] – a context-aware application which helps visually impaired people to orient themselves in new, unknown environments, thereby enabling them to lead a more independent life. Furthermore, the digital augmentation of real-world objects can help to compensate for deficiencies of cognitively challenged people. An augmented jigsaw puzzle we developed, for example, may not only help children or people with cognitive deficiencies to solve complex puzzles, letting them partake in the experience and feeling of achievement, but it also facilitates to match the capabilities of unbalanced players by providing aids to the "weaker" players [Boh04b].

16.5.2. Soft vs. Hard Augmentation

Intelligent interfaces and the concept of ubiquitous information access are often seen as key developments for bridging the digital divide caused by the situation where different sections of the population have different abilities to participate in the information society (see also Sect. 16.3.4). One way of preventing the further development of a digital divide lies in ensuring that an augmentation of real-world artifacts with information processing and communication capabilities does not become an end in itself. Instead, the expected benefits should, from the beginning, be weighed against potential negative side-effects. It may be advisable to deliberately stop the augmentation process at some point before the original fundamental qualities and characteristics of the augmented physical object are threatened to be lost. Then the result is a *soft* augmentation that preserves knowledge sustainability: the original object used to work this way and it still does after the augmentation. This is opposed to a *hard* augmentation that fundamentally alters the mode of usage or properties of an augmented object. A soft augmentation has the advantage that it empowers the user to deliberately opt out and revert to the classical unaugmented utilization of the object if desired. Another advantage of a soft augmentation is that the usability of the augmented object is still sustained even in case of a technical failure of the augmented functionality. If, however, the inherent qualities and functionality of the original object are irrevocably changed, the usability of the augmented object may largely depend on the availability and proper functioning of the technologies used in the augmentation process [Boh04b].

16.5.3. Open Ubiquitous Computing Systems

Another solution to the problem of creating a digital divide is to adhere to the *open-world assumption* that constitutes a fundamental property of ubiquitous computing systems (see Sect. 3.3.9). By designing ubiquitous computing systems to be truly open, it is possible to share the potential benefits with marginal and unprivileged user groups. For instance, the concept of super-distributed smart entities, which we introduced earlier in Sect. 9.4.2, can serve as a design principle for building open, user-centric service infrastructures. For instance, the location-aware services and applications we described in this dissertation are open to the general public, explicitly including elderly persons or people with disabilities who often suffer from disadvantages caused by their physical or mental deficiencies (see Sect. 9.4.3.

16.5.4. Coping with Novel Dependencies

Despite their potential benefits, the application of ubiquitous computing technologies for the augmentation of physical objects and for the realization of ubiquitous information environments is very likely to induce new societal and technological dependencies. In particular, as the number of smart devices and interacting objects in our environment increases, the technical dependability of the thus provisioned services becomes an important issue. Traditionally, a user explicitly works with dedicated computer equipment which often consists of reliable quality components. With the expected coming of the ubiquitous information society, however, users find themselves suddenly acting right in the middle of a computerized smart environment. They have to cope with being caught in a crossfire of mass-produced,

low-cost smart artifacts and spontaneously interacting objects, each of which is prone to malfunctions due to technical defects or depleted batteries, for example.

16.5.5. Technical Contributions and Solutions

The introduction of new dependencies and the potential social problems they induce raises the question whether there exist technical solutions to counter such difficulties, such as employing physical redundancy, for example. However, as incorporating computing and communication technology into everyday artifacts requires small form factors and minimal energy consumption, it is often impracticable to employ hardware redundancy on the single devices to increase the fault tolerance and robustness of smart object infrastructures.

We think that one possible answer to these challenges can be found in alternative, more user-centered concepts and mechanisms that overcome service interruptions and device failures. For instance, we have presented systems that make us of an *explicit diversification of system functions*. By providing fully independent ways of carrying out the same task, preferably based on separate sets of system resources wherever feasible, the user's dependence on individual technologies and services can be reduced.

Further, as low-cost individual devices are prone to technical defects and malfunctions, the concepts we presented earlier such as the *super-distribution of smart entities*, *localized interaction with cooperative smart objects*, or the *instant personalization of handheld devices*, help to increase the availability of services and device functionality with regard to the individual user, at his or her particular location. We strongly believe that the explicit shaping of ubiquitous computing infrastructures around the individual user, the tailoring of ubiquitous computing systems to suit the needs of inexperienced individuals and advanced users alike, constitute paramount challenges and design goals of ubiquitous computing.

16.6. Conclusion

"Everything will be connected to everything else," but "no one has any idea what all those connections will mean" [Luc99]. This criticism on the one hand indicates a perceived lack of focus when it comes to ubiquitous computing applications, but also points at a deficiency in terms of understanding the consequences of deploying ubiquitous computing systems in the real world. Fundamental questions are how we will use "smart things" in our everyday lives, when we should switch them on or off, what smart things should be permitted to hear, see, and feel, and whom they should be allowed tell about it. Whether the consequences of ubiquitous computing systems concern the protection of personal data, the implications for the macroeconomy, or social acceptance – developers of ubiquitous computing systems can profit greatly from a careful evaluation of the consequences of such technology within the framework of established concepts from the fields of sociology, economics, and jurisprudence.

Although predicting the future is difficult, if not impossible, the above discussion allows us to guess at a few of the possible implications of a wide-scale use of ubiquitous computing technology. However, in order better to understand how far emerging ubiquitous computing systems can and should influence our everyday

lives, it is important to identify and address the great challenges of technical and social change, and of their environmental sustainability. Our goal is to help steering the development in ubiquitous computing in a direction that has more in common with Weiser's optimistic vision of the 21st century than with the depressing mix of consumer terror and police state conjured up by Steven Spielberg in his movie "Minority Report" [Dic56]. The user-centric dependability challenges we identified and described in this dissertation, and the solutions and concepts we presented to address these dependability challenges, contribute to achieving this goal.

Acknowledgments

The author wishes to thank Vlad Coroama, Marc Langheinrich, Friedemann Mattern, and Michael Rohs for the many fruitful discussions and helpful suggestions about potential social dependability challenges and implications of ubiquitous computing.

Part V.

Summary and Conclusion

17. Summary and Conclusion

In this final chapter, we recapitulate and summarize the major contributions of this dissertation, and round off our work with a brief conclusion.

We would like to point out that the major contributions of this thesis have also been published in journals, conference and workshop articles as well as research deliverables, most notably [BCL+03a], [BCL+03b], [Boh03], [BV03], [RB03], [BCL+04a], [BCL+04b], [BM04], [Boh04a], [Boh04b], [CBM04], [Boh06], [Boh07a], [Boh07b], and [RABB07]. Furthermore, there are several student projects and Master's theses within the scope of this dissertation which also dealt with concepts presented in this thesis. Of particular relevance are [Kai01], [Sch02], [Maz03], [Bär04], [Geg04], [Pir04], [Stu04], [Bur05], [FL05a], and [Opr05].

17.1. Main Contribution

In this dissertation we provided a systematic study of the problem of *user-centric dependability* in the context of ubiquitous computing, which so far has not been researched deeply in the research community.

We first reviewed conventional dependability issues and methods for achieving fault tolerance in the domain of distributed computing, and described the vision and background of ubiquitous computing. Then we performed an analysis of fundamental characteristics of ubiquitous computing in general, which showed that ubiquitous computing systems feature *challenging technical boundary conditions* such as a high degree of distributedness, heterogeneity of resources, and system dynamics, and an unprecedentedly high level of *user-centricity*.

Based on our initial analysis, we discussed the application of conventional dependability methods in ubiquitous computing, and we investigated different means of redundancy that can be employed for achieving fault tolerance in highly distributed and dynamic ubiquitous computing systems. We also showed that concepts for *dependability* in ubiquitous computing have to be *user-centric* and designed to deal with novel types of faults.

17.2. Individual Contributions

We focused on the investigation of dependability methods for two principal fields of research in ubiquitous computing, which are human-computer interaction, and context- and location-aware computing.

17.2.1. Concepts for Dependable Human-Computer Interaction

With regard to human-computer interaction, we showed that the *accessibility* of user interface devices poses a novel dependability challenge. Addressing this challenge, we described two concepts that enable users to harness the diversity and multitude of devices found in ubiquitous computing environments, and which reduce the users' dependence on individual devices or technologies: Firstly, the concept of *input/output diversification*, which enables users to make use of diverse user interface devices for controlling and interacting with a surrounding smart environment. Secondly, *instant personalization*, a concept that enables users to personalize arbitrary mobile devices on demand, thus making personal user devices interchangeable and overcoming the barriers of exclusive personal use and ownership that oppose the sharing of ubiquitous handheld devices among larger user groups.

17.2.2. Concepts for Dependable Context-Aware Computing

With respect to context-aware computing, we showed that the dependability of mobile devices and applications can be improved by exploiting the redundant resources found in the immediate vicinity (*locality*). Concretely, we presented concepts that enable resource-limited mobile devices to achieve fault-tolerant operation in the case of temporary unavailability of diverse computing resources, based on localized redundancy acquired in an ad hoc fashion from the local computing context: *localized cooperation and resource sharing* and *redundant multi-sensor data fusion*.

The concept of *localized cooperation and resource sharing* enables mobile devices to tolerate resource shortages during operation by acquiring additional resources through localized cooperation with volatile *smart everyday objects*, or by using the resources and services provided by a dedicated physical infrastructure consisting of *super-distributed smart entities*. In the smart-object-based approach, we presented a generic middleware layer that enables mobile devices to exploit the memory storage and communication capabilities of proximate computerized entities for fault-tolerant data dissemination and communication. Here ubiquitous computing devices take on the role of local service providers and communication gateways. In the case of super-distributed smart entities, we showed how a dense and highly redundant distribution (i.e., *super-distribution*) of computerized entities over physical spaces can be turned into a powerful infrastructure for the creation of reliable and highly available location-dependent services. In the process, we described the design of a fault-tolerant service middleware architecture. We also presented a concrete realization of the concept based on radio frequency identification as an enabling technology. As a proof of concept, we presented and experimentally evaluated a prototypical implementation of our middleware.

We further showed that for mobile devices and applications that rely on knowledge about their current contextual situation for providing *situation-aware* services, it is essential that the process of context sensing and inference itself is robust and dependable. We elaborated the concept of *redundant multi-sensor data fusion*, which exploits redundancies in sensing capabilities for maintaining a minimum quality of service even if individual sensors temporarily fail or become spontaneously unavailable while operating in highly dynamic ubiquitous computing environments. As a case-study, we presented the iPOS system, a redundant multi-sensor data fusion

architecture that enables autonomous mobile clients to determine their geographic position and/or symbolic location in an adaptive and fault-tolerant manner.

Finally, we demonstrated how the above approaches can be profitably combined into a single hybrid system. For that, we presented and evaluated a positioning system which combines light intensity measurements by means of a solar cell with position information obtained from a super-distributed RFID tag infrastructure for enabling fault tolerance, self-calibration, and adaptability.

17.2.3. Social Dependability Challenges

We showed that the pronounced user-centricity of ubiquitous computing systems and applications leads to a number of *social challenges and implications* that reach beyond mere technical dependability issues. We discussed a number of fundamental social challenges with regard to *reliability* in the broadest sense, including the issues of *delegation of control*, *social compatibility*, and *acceptance*.

17.3. Conclusion

User-centric dependability is a novel challenge closely connected to the paradigm of ubiquitous computing, where the user finds him or herself pushed from the fringes into the center of computing systems. In this dissertation, we described and investigated various facets and challenges of user-centric dependability, and we motivated the relevance of this emerging field of research in the context of mobile and ubiquitous computing. We further presented a number of concepts that we think are appropriate to address some of these user-centric dependability challenges. In addition, as proof of our concepts, we described and evaluated several prototypical demonstrators and reference implementations. Finally, the author wishes to express his hope that this dissertation may serve as a valuable starting point and catalyst for future research in the field.

Part VI.

Appendix

A. Dependability

In the following, we give a short overview of the vast field of dependability in distributed computing. In large parts, we follow the explanations and observations of Jalote, according to his book on "fault tolerance in distributed computing" [Jal94].

A.1. Definition

Dependability is conventionally defined as the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers [Lap85, Lap92a]. In this context, the service delivered by a system is its behavior as it is perceived by its users, while the user may be human or another physical system.

Laprie further defines fundamental attributes which emphasize different, complementary *properties* of dependability. Firstly, these attributes of dependability serve to express the properties that are expected of a dependable system. Secondly, they allow the system quality resulting from the impairments and the means opposing them to be assessed. The most significant attributes of dependability are *reliability*, *availability*, *safety*, and *security* [Lap85, Lap92a]. Reliability deals with the property of *continuity of service*, availability with *readiness for usage*, safety with *avoidance of catastrophic consequences*, and security with *prevention of unauthorized access and/or handling of information.*

A.2. Terminology

A.2.1. Distributed Systems

When modeling processes in the real world, one often finds that different sub tasks have to be carried out in different locations, looking at production chains in manufacturing plants for instance. Hardware and software may therefore be physically distributed and operate self-sufficiently on local tasks. For a distributed system to be in a position to achieve a common global task efficiently, the different distributed pieces of hardware have to be able to reliably communicate to be in a position to cooperate and coordinate the different sub tasks of the superordinate process. But although the data, software, and hardware of a distributed system may be distributed across several locations, the distribution itself and the complexities involved with it are typically masked from the user, to whom it appears as if the system components were united, acting as a single entity.

Traditionally, distributed systems are viewed in two ways, either as defined by its physical components, or as defined from the point of view of processing or computation. In the first case, we speak of the physical model of the system, and in the latter case of the logical model.

The *physical network* of a distributed system consists of many computers, which are also called nodes. The nodes are geographically at different locations, and connected with each other by means of a communication network, through which they communicate with each other by exchanging messages. Further, all nodes are autonomous entities. Each node is equipped with a processor, which has some private volatile memory, a private clock that is used for coordinating the internal execution of instructions, a network interface through which the node is connected to the communication network, and software that governs the sequence of instructions to be executed on the node. These components of a node are considered to be *atomic*. Fault-tolerance mechanisms in distributed systems aim at masking the failure of some of these components to prevent the entire distributed system from failing. Often a distributed system is modeled as consisting of nodes and of a communication network as the basic components. In this case, the main component failures that have to be addressed by fault-tolerance mechanisms are the failure of a node and the failure of the communication network.

The *logical model* concentrates on the applications viewpoint of distributed systems: a *distributed application* consists of a finite number of concurrently executing processes that cooperate with each other to perform some task. A process is defined as the execution of a sequential program, which in return is a list of statements or instructions. Further, concurrent processes can either be executed on a single processor, or in parallel on different nodes in the system, which is the more interesting case for a distributed application. The interactive behavior between concurrent processes can be categorized as independent, competing, or cooperating. *Independent* means that the processes work on disjoint sets of objects (hardware resources and data), which is conceptually equal to physically separate sequential processes. Concurrent processes are *competing* if they share resources but at the same time do not exchange information between themselves. This corresponds to a set of independent processes. *Cooperating processes* are characterized by an exchange of information which is either based on message passing or on using shared data objects. In the context of distributed systems as they are traditionally seen, only message passing is possible, as no shared data is allowed, and processes are usually considered to be cooperating. From the applications point of view, the underlying network is treated as a fully connected network, assuming the physical network is connected. This implies that a message can be sent from one node to any other node, which is supported by suitable communication protocols. Consequently, the network topology is not considered at this level. The logical connection between any two processes which interact by means of message passing is called a *channel*. A channel is assumed to have infinite buffer, to be error-free, and to deliver messages in the order that they have been sent (which is ensured by the underlying communication protocols).

In the logical model, the performance of the system is usually described with respect to applying time bounds [MS92]: If a correctly working system always performs its intended function within a finite and known time bound, it is called *synchronous*; otherwise, it is called *asynchronous*. Similarly, a *communication channel* is *synchronous* if maximum message delay is known and bounded; otherwise it is called *asynchronous*. A *processor* is *synchronous* if the time for executing a sequence of instructions is finite and bounded, otherwise it is called *asynchronous*. The main advantage of synchronous systems is that the failure of a component can

be detected by means of a "timeout", which means that a response is not obtained within some defined time bound.

Failures that occur in the physical system can cause the failure of components in the logical system: If a physical node fails, it may cause the failure of some processes, which can be considered as logical nodes. Similarly, the failure of communication lines in the physical network may cause the failure of logical channels. According to Jalote, the primary goal of fault tolerance is to "preserve some properties in the logical model despite some failures in the physical model" [Jal94].

A.2.2. Faults and Errors

The dependability of a system is threatened if *faults* in components of the system occur. Faults are the cause for errors which in return are *liable* to lead to subsequent failure. A *failure* manifests itself by a system behavior that is not compliant with the specifications, the latter being an agreed description of the system's expected function and/or service.

Faults and their sources are extremely diverse. They can be classified according to their nature (accidental or intentional), their origin (phenomenological cause: physical phenomena or human-made; system boundaries: internal or external; phase of creation: design or operational), and their persistence (permanent or temporary). Note that a physical or human made external fault is considered a design fault, because it should have been foreseen and considered during the design phase.

Laprie defines an *error* as that part of the system state which is liable to lead to subsequent failure. An error is always the consequence of a fault, but whether it actually leads to a failure depends on three major factors [Lap92b]:

- System composition, especially with respect to the nature of existing redundancy, which may either have been introduced *intentionally* to provide fault tolerance, or which may have been introduced *unintentionally* and which may lead unexpectedly to the same results as intentional redundancy.
- System activity, during which an error may be overwritten before it creates any damage.
- The definition of a failure from the user's viewpoint, which may vary considerably for different users, because what one user considers a failure may be a bearable nuisance for another.

Consequently, faults, errors and failures can be considered the *impairments* to the dependability of a system, potentially leading to an undependable system on whose services reliance cannot or will not any longer be placed.

A.2.3. Classification of Faults and Failures

In distributed systems, the major components are processors, communication links, clocks, nonvolatile storage, and software. In the system model, these components are seen as atomic. Fault tolerance normally aims at tolerating failures of these components, with the major goal of ensuring the continuity of service on behalf of the distributed application even in the presence of failures.

A well established method of classifying the faults that may occur in distributed systems is based on how the faulty component behaves when it fails is to distinguish the following four failure categories [Jal94]:

1. *Crash fault:* A fault that effects that the component halts or loses its internal state, without undergoing any incorrect state transitions.

2. *Omission fault:* A fault that results in a component not to respond to some inputs.

3. *Timing fault:* A fault that causes a component to respond either too early or too late. This fault is also called *performance fault*.

4. *Byzantine fault:* An arbitrary fault which causes the component to behave in a totally random manner during failure. This category subsumes the so-called *incorrect computation fault*, where a component does not have any timing fault but produces an incorrect output to the given inputs.

The four categories form a hierarchy where the category with the higher enumeration number contains all categories with lower numbers. Consequently, the crash fault is the simplest and most restrictive, and the Byzantine fault the most complex and general fault.

Component	Crash	Omission	Timing	Incorrect Computation	Byzantine
Processor	++			+	+
Communication Network	+	+	+	+	+
Clock		+	++		+
Storage Media	+	+	+	+	
Software	+	+	+	++	+

Table A.1.: Major distributed system component categories and the fault categories by which they are commonly affected. Faults that are common for a component are indicated by '+' or '++' (the latter signifying the most prominent fault type for the category if applicable)

Typically a crash fault is assumed for a processor that stops, or else a Byzantine fault.

A communication network may show any of the fault types, such as crash faults if it does not deliver messages, omission faults if messages get lost, timing faults if messages are significantly delayed, incorrect computation faults if messages are corrupted, or Byzantine faults if it behaves in a totally arbitrary fashion.

The most common fault of a clock is that it runs too slow or too fast (timing fault), but it may also stop and always show the same time (omission fault), or display a totally arbitrary behavior (Byzantine fault).

Storage media may suffer from crash faults (data cannot be read/written any more), timing faults (data arriving late), omission faults (some data inaccessible), and incorrect computation faults (corrupt data).

Software components may show any kind of fault behavior, but the most interesting fault type is the incorrect computation fault, resulting in wrong computations on behalf of the software, which are typically also called *software design faults*.

An overview of distributed system components and the fault categories by which they are commonly affected is shown in Table A.1.

There are also notions for describing the behavior of a component in response to a detected fault. A system component is considered *fail-safe* when it places itself in a safe operating mode in the event of a failure, or *fail-soft* if it continues to provide partial operational capability in the event of certain failures [IoEEE90]. In case a failure occurs, the behavior of an affected component is called *fail-stop* behavior if the component stops executing without performing any incorrect actions, losing its internal state and the content of any associated volatile storage data, and if the failure of the component can be detected by other components. The importance of fail-stop behavior of components for the realization of fault-tolerant systems has been underlined by Schlichting et al. in their work on so called *fail-stop processors* [SS83], which embody the described fail-stop characteristics. In this context, they have also defined a *k-fail-stop processor* as a computing system that still behaves like a fail-stop processor as long as no more than k components of the system fail.

A.2.4. Classification of Fault-Tolerance Methods

The development and maintenance of dependable computing systems calls for suitable *means*. According to Laprie, dependability can be achieved by the combined utilization of a set of methods that can be classified into the following four groups:

- *Fault Prevention*: methods to prevent the occurrence or introduction of faults;
- *Fault Tolerance*: methods to provide a service complying with the service specification even in the presence of faults;
- *Fault Removal*: methods to reduce the presence (in terms of number or seriousness) of faults;
- *Fault Forecasting*: methods that estimate the present number, the future incidence, and the consequences of faults.

Fault prevention and fault tolerance are used to *provide* a system with the ability to deliver a service complying with the specification, thus constituting "dependability procurement". In contrast, fault removal and fault forecasting are more concerned with the "validation" of the dependability of the system.

Fault prevention tries to eliminate as many sources for faults as possible before the system is put in regular use without the deployment of redundancy. Fault tolerance, in contrast, uses *protective redundancy* [Jal94] to automatically mask failures and to avert system failure in case some components fail. Note that fault tolerance requires redundancy – without the employment of *redundancy* of some kind, a system cannot become fault-tolerant [Gär99].

A.2.5. Safety and Liveness

For assessing the usefulness of dependability measures applied to distributed systems, usually two major classes of system properties describing the system behavior are analyzed, which are called safety and liveness [Lam77].

Formally, a *safety property* is characterized by specifying when an execution e is not safe for a property p (which means it is not contained in a safety property p). In other words, if $e \notin p$, then there must be an identifiable discrete event within e that prohibits all possible continuations of the execution from being safe. So a safety property is expressed by a set of "legal" system configurations, and a distributed program that has been proven to be safe will always stay within this set of safe states. In this context, a property of a distributed program is defined as a set of system executions, whereby an execution of a distributed program is given by an infinite sequence $e = c_0, c_1, c_2, \ldots$ of global system configurations. Consequently, a distributed program always defines a property in itself, which is the set of all system executions that are possible from its starting configuration. A specific property p is said to hold for a distributed program if the set of sequences defined by the program is contained in p. The informal interpretation of a safety property is that it states that some specific unwanted and irremediable "bad thing" *never happens* within a system [Gär99].

A *liveness property* informally states that some "good thing" will *eventually happen* during system execution [Gär99], thus capturing notions of "progress". Formally, a liveness property is defined as a property for which every partial execution is live [AS85], with a partial execution of a system being considered live for property p if and only if it can be extended to still remain in p (i.e., further system configurations can be added to the execution space without violating the safety property). According to Gärtner, the most common example of liveness in distributed systems is termination [Gär99], where a certain goal (the "good thing") is reached eventually (without making a forecast as to when this will actually be the case).

A.2.6. The Consensus Problem

Fundamental to many basic fault-tolerance mechanism for distributed systems is the *consensus problem*: it lies at the core of protocols handling synchronization, reliable communication, resource allocation, task scheduling, reconfiguration, replicated file systems, sensor reading, and other functions [BDM93].

The basic and simple idea of consensus is to share information among a group (or population) of processing elements (PEs), and if possible, to do so in a fault-tolerant manner. Even if a part of the PE population is acting faulty or maliciously, the fault-free PEs still should be able to consistently agree on and produce correct results. So consensus is required whenever several components or processes have to decide on a correct result in the presence of faulty results or malicious components/processes.

In distributed operating systems, consensus procedures are used on different layers. Firstly, if time is to be used for detecting untimely messages of an underlying unreliable communication medium, for diagnosing faulty processors, or for agreeing on the correct sequence of computations and a correct result, consensus about a particular time value needs to be established among the fault-free PEs. Secondly,

with this knowledge it is then possible to establish reliable communication links between two arbitrary processors, which involves consensus on the set of information and the transmission order of that set. Thirdly, by means of established reliable communications, consensus on the diagnosis of the system can be reached by all fault-free processors, which is required for a consistent reconfiguration of the overall system after a fault [BDM93].

Consensus among several processors can be reached by means of n-modular redundancy (NMR): With NMR, n PEs perform the same task, which enables the masking of t faulty PEs, with $n > 2t+1$, by taking a majority vote of the n results. However, this is achieved at a great cost of resources while the throughput (jobs per unit time) is limited to the throughput of a single PE. By identifying the fault-free PEs in a reliable fashion, it is possible to increase the throughput of the n-processor system, since the faulty processors can be ignored in the task scheduling process in favor of the fault-free ones, removing the need to mask them instead. Thus, if a diagnosis of faulty processors can be performed reliably, the performance gain factor of the system over the original NMR technique is given by the number of fault-free PEs [BDM93]. The diagnosis itself, that is the detection and location of faulty processors and the dissemination of that information among the fault-free processors, has to be correct, too. However, achieving a reliable diagnosis is often very costly, since the fault status of a system may be obsolete as soon as it is calculated, as faults may trigger recovery procedures, for example, which means that NMR techniques may still be needed if the costs for diagnosis and recovery are too high.

The implementation of NMR requires a voting mechanism that combines the n results into a single output, and the reliability of this mechanisms is obviously related to reliability of any process using the output. Therefore, if multiple processors rely on the output of the NMR system and the subsequent computations of these processors must be consistent, then every processor must be able to agree on the output of the NMR system [BDM93].

Obviously, the reliability of any process that uses this output is directly related to the reliability of the voting mechanism. If a single process is using this result, then it is sufficient for it to act as its own voter as the voting process will fail exactly when the process fails. But when (1) multiple processors rely on the output of the NMR system and (2) their subsequent computations must be consistent, then every processor must be able to agree on the output of the NMR system. Since a single point of failure is unacceptable for the voting procedure, the voting mechanism itself must be distributed. This leads to the so-called *Byzantine Generals Problem*, or *Byzantine agreement*, which explores solutions of the consensus problem given the need for a reliable and fault-tolerant distributed voting process [BDM93].

A.2.7. Byzantine Agreement

A fundamental challenge of reliable distributed computer systems is the handling of malfunctioning components that give conflicting information to different parts of the system. It has been shown that the challenge of reaching an agreement among distributed computing entities upon a single correct value in the presence of malicious or faulty entities is equivalent to the so-called *Byzantine Generals Problem* [LSP82]. Abstractly, the agreement problem of distributed entities or

processes can be mapped to the problem of a group of distributed Byzantine generals who have to agree on a common battle plan. This problem is known under the name of *Byzantine agreement*. The generals are located on different sides of the enemy city and can only communicate by means of messengers. However, one or more of the generals may be traitors who try to sabotage the battle plan by confusing other generals about the actual battle decision. Now the problem is to find an algorithm that ensures that all loyal generals will reach agreement even in the presence of traitors. It could be shown that this problem is solvable using only forgeable oral messages (e.g., the messenger can lie about the true content of the oral message) if and only if more than two thirds of the generals are loyal. With unforgeable written messages, the Byzantine Generals Problem is even solvable for any number of generals and possible traitors [LSP82].

The original Byzantine Generals Problem can be weakened by allowing the processes to agree upon an incorrect value if a failure occurs, which is consequently called *Weak Byzantine Generals Problem*. Here an agreement can still only be reached if fewer than one third of the processes are corrupted. However, unlike the original problem, it has been shown that an approximate solution for the Weak Byzantine Generals Problem exists that can tolerate an arbitrary number of failures [Lam83].

A.2.8. Reliability and Availability

The main goal of any fault-tolerant system is to increase the reliability and availability of a system. In practice, these two attributes are often expressed by means of statistical terminology. For defining reliability, the lifetime of a system, which is equivalent to the time to failure of the system, is considered as a random variable. According to Jalote, the *reliability* of a system at a time is defined as the "probability that the system is operational at that time instance" [Jal94]. The *expected life* of a system is given by the expectation of the reliability function, which is also called *mean time to failure* (MTTF). Since faulty components are generally repaired or replaced, a system may either be in one of two states, which are "working" or "under repair". Based on this observation, *instantaneous availability* of a system at a time can be defined as the "probability that the component is working correctly at that time". The only difference between this definition and the definition of reliability is the consideration of system repair – if there was no repair, the two definitions would be equal. The expected time it takes to repair a faulty component is called mean time to repair (MTTR). *Steady state availability*, which is generally just called the *availability*, is defined as the "limit of instantaneous availability as time tends to infinity", which represents the fraction of the time the system is operational. The thus defined availability of a system can be expressed by means of MTTF and MTTR as MTTF/(MTTF+MTTR). Obviously, the availability of a system is independent of the distributions of the lifetime and repair times.

A.3. Hardware Fault-Tolerance

The lower level of computing systems is represented by the hardware on which the system software and processes are executed, while system software and applications

are part of the higher levels. On the lowest hardware level, we find basic units such as gates, transistors, or integrated circuits (ICs). Higher hardware levels contain more advanced units such as processors or arrays of memory cells.

As a rule, faults that lead to the failure of a hardware component have a physical cause. Physical failures of hardware can already occur during fabrication (e.g., faulty hardware parts, break in connections, short-circuits between lines, improper doping, impurities in packaging, etc.), or they may occur with the passage of time (e.g., short-circuits, breaks, shift in threshold voltages).

A.3.1. Hardware Fault Models

The effects of physical failures can be specified by means of an abstract *fault model*, which maps the manifold possible physical failures to a limited number of abstract higher-level faults. In the following, some common fault models are listed that differ in the level of abstraction from the underlying hardware:

- *Gate-level fault models* are very frequently used in hardware fault-tolerance to describe detailed physical failures with respect to the gate-level structure of a circuit. An example is the classical *stuck-at* fault model. This fault model assumes that physical failures lead to the permanent state of 0 or 1 of the affected logic gates. This captures bonding failures and circuit breaks, which are considered the most common failures in small ICs [Jal94]. The *stuck-open* fault model captures an observed "memory effect" of circuits, which is usually due to a break in a line of CMOS transistors, and which may cause combinational circuits to act like sequential circuit, for instance. If a transistor permanently conducts, this is called a *stuck-on* fault. Further gate fault models are the *bridging fault model* describing short-circuits between adjacent lines, and the *delay fault model*, describing a delay of a gate or a path, which in return may cause a circuit to produce incorrect logic output values.

- *Function-level fault models* are modeling failures at the level of functional modules, which includes functional blocks (N input lines are logically combined and the result is provided on M output lines), multiplexers, iterative logic arrays. Ideally these fault models include the effects of physical failures of faults at the gate level.

- *Memory fault models* for the main memory (Random Access Memory, RAM) of computer systems addresses faults such as memory cells that are stuck at 0/1, couplings between cells that makes them change values together, or changes in the memory state of some cells that are affected by changes in other memory cells.

A.3.2. Basic Techniques

Some of the most commonly used techniques to support hardware fault-tolerance are triple/N modular redundancy, dynamic redundancy, coding, and self-checking circuits.

Triple Modular Redundancy

Triple modular redundancy (TMR) was first suggested by Von Neumann and is considered the most commonly known technique for hardware fault-tolerance. Its characteristic is that a hardware unit is rendered fault-tolerant by triplicating it: three units work in parallel on the same task, and the three outputs are processed by a voting element, which performs a majority vote. Thus, TMR can completely mask the failure of one hardware unit, without explicit need of error detection and recovery mechanisms. The TMR scheme depends critically on the voting element. Therefore, the voting element is typically a simple circuit with low complexity, which can be built in a highly reliable fashion. A more general approach is to replicate the hardware unit N times instead of 3 times, with $N > 3$, which is called *N modular redundancy* (NMR).

Dynamic Redundancy

Another basic technique for hardware fault-tolerance is called *dynamic redundancy*. Here a system consists of several units of which only one is operating at a time, while the others are serving as spare units ("spares") that can be "switched in" by a switching circuit when a faulty circuit is detected (which is then "switched out") [Lal85]. A dynamic redundancy system is further called *cold-standby system* if its spares are only powered up on demand ("cold" start) when needed to replace the faulty active component which is then powered down. In a *hot-standby system*, all units are active and operating simultaneously, and their outputs are compared. If the outputs match, then an arbitrary output is chosen. Otherwise, the faulty component is detected and disabled, reconfiguring the system that it uses the results from a non-faulty component [Lal85]. The most common hot-standby system is the so-called *duplex* system, where two units are executed in parallel, and the output results are continuously compared. In case of an output mismatch, diagnosis routines are launched to locate the fault, upon which system can be reconfigured.

The main difference in dynamic redundancy compared to the TMR (or NMR) approach lies in the way a faulty unit is detected and removed. TMR is particularly suited for transient faults were no components have to be replaced, and performing fault detection is not necessary for achieving fault tolerance. In contrast, a key challenge of the dynamic redundancy approach is to detect the failure of a unit, which commonly is achieved by means of periodic tests, self-checking-circuits, and watchdog timers [Jal94].

Coding

A further important means for achieving hardware fault-tolerance (and reliable communication) is *coding*. Here the basic idea is to add redundant check bits to the information such that errors in some bits can be detected or corrected. This provides a mechanism for performing structural checks for verifying the structural integrity of data. The process of adding check bits is called *encoding*, and vice versa, the reverse process of extracting data from the encoded data is called *decoding*. Some commonly used codes in hardware fault-tolerance are Hamming codes, cyclic redundancy codes, and Berger codes.

Hamming codes use a more general method of parity bits: multiple parity bits are

added in a way that each parity bit determines the parity of a well-defined subset of information bits by means of the exclusive-or (XOR) operation. Hamming codes can detect and correct bit errors, the number of which depends on the number of parity bits used and on the Hamming distance of the code. The *Hamming distance* of a code is defined as the minimum number of bit positions in which any two code words differ. Further, for a Hamming code with Hamming distance d, which can detect D bit errors and correct C bit errors, the relation $d = C+D+1$ is always true, with $D \geq C$. A code using a single parity bit, for example, has a Hamming distance of 2, and can only detect single bit errors. A significant property of hamming codes is that the overhead given by the number of parity bits decreases relatively as the length of the protected information word is increased. A typical field of application for Hamming codes is the area of semiconductor memories.

Cyclic redundancy codes (CRCs) are applied to blocks of data rather than to independent words. CRCs are characterized by a so-called *generator polynomial* $G(x)$ with some degree k. For *encoding* a data block, this data block is first transformed into a polynomial: this is achieved by adding term x^j for all bit positions j of a bit string where the bit has the value 1. Then the thus obtained polynomial is divided (modulo 2) by the generator polynomial. This results in a final remainder of $k+1$ bits which forms the CRC checksum and is added to the data block. To verify the integrity of a data block (CRC *check*), the data bits are again divided by the generator polynomial, and the remainder is compared to the previously calculated CRC checksum. A mismatch signifies that one ore more bit errors have occurred. *Decoding* is achieved by simply removing the $k+1$ bits of the CRC checksum from the data block, if no errors have occurred. CRCs can detect all single bit errors, and all burst errors of a length less than k. In addition, all other bit errors that are not divisible by the used generator polynomial can be detected.

Berger codes count the number of zeros (0s) in the data word and append this number a checksum to form the code word. Given a code word size of k bits, this requires $log_2(k)$ extra bits. Berger codes are known to be optimal for the detection of unidirectional errors, where all the erroneous bits change from 1 to 0 or vice versa from 0 to 1, even if these errors affect the check bits, given that the original information and the check bits can be separated. There is also an alternative version of the Berger code that counts the 1s instead of the 0s.

Self-Checking Circuits

Performing error detection by using a coding scheme requires the presence of a so-called *checker* which verifies whether an encoded word is valid or not. A typical usage scenario for the application of a checker is to verify the coded output of functional circuits. However, if the checker itself is faulty, it may happen that errors in the coded output are not detected. The goal of self-checking circuits therefore is to detect errors in the output of the functional circuit as well as to detect faults in the checker itself. Although there are no general techniques for designing self-checking checkers [Jal94], some techniques have been proposed for the synthesis of self-checking combinational and sequential circuits [JW93].

A.4. Fundamental Dependability Concepts

A.4.1. Basic Building Blocks

Jalote divides classic fault-tolerant distributed systems into different levels of abstraction [Jal94]. On the lowest abstraction layer, right on top of the underlying distributed system, *basic building blocks* are described that are frequently needed for the realization of more complex fault-tolerant services. These building blocks include abstractions of fail-stop processors [SS83] (i.e., processors that simply stop functioning and which do not perform any incorrect actions when a failure occurs), stable storage (i.e., the content of a stable storage is not lost or corrupted in the presence of failures) [BJM+91, SS83], reliable communication [BJ87, Bir93], synchronized clocks [LMS85], and failure detection [ACT00]. The next level are the abstractions of reliable and atomic broadcast [GS97, KHTK00] and multicast [SBS91], which are usable for supporting one-to-many communication and which in return require the basic services of reliable point-to-point message delivery and fail-stop processor abstractions. These two levels provide building blocks for higher fault-tolerant services. The simpler ones among these are *consistent state recovery* [Koh81, HR83, SY85] to reach a consistent system state [BG95] if any error occurs, and *atomic actions* [Ree83, Rom01] to ensure the atomicity of operations and transactions [LS81] even in the presence of failures. Jalote defined a user-defined action as "a sequence of primitive operations or steps which are executed indivisibly by the hardware". Consequently, a user-defined atomic action either completes fully, or it appears that the system has not performed any action at all, which is achieved by recovery mechanisms that undo or redo the operations that are part of the atomic action in case a failure occurred. Established recovery concepts are checkpointing, and commit protocols.

A.4.2. Data Replication and Resiliency

An action is considered *resilient* to failures if it can still be completed successfully even when failures occur in the system, thus masking those failures. This is different to the concept of atomic actions where an action is rolled-back (or repeated) in the presence of failures. Naturally, if a data object is only residing on a single node, nothing can be done to prevent an action to fail if the respective node fails. Therefore data objects have to be replicated on multiple nodes to achieve redundancy, which is a prerequisite for tolerating failures. So while data replication can be used to achieve resiliency against failures, it also introduces new problems of consistency and replica management. One requirement is that concurrent actions performed on replicas is equivalent to a correct execution on non-replicated data, which is equivalent to a serial execution of actions on non-replicated data. The correctness property is therefore also called the *one-copy serializability* criterion, and the corresponding methods to enforce the criterion are called *replica control algorithms*. Since the one-copy serializability criterion also requires that different copies of a data object are in a mutually consistent state, so that each user action gets the same view of the data object, replica control mechanisms are also called *consistency control mechanisms*.

Replica control mechanisms can be divided into so called optimistic and pessimistic approaches. Optimistic approaches have the optimistic hope that opera-

tions that are executed in different network partitions will not conflict, and if they do, that these conflicts will be resolved at a later point in time. Pessimistic approaches prevent inconsistencies from occurring by limiting access to data [Jal94].

An example for an optimistic approach is the use of so-called *version vectors*, where files are treated as basic data objects used for replication. Each copy of a file is associated with a version vector, which keeps track of updates originating at different nodes that were performed on the respective copy, and which can be used to resolve different copies if the version vectors do not conflict. If partitioning occurs and conflicting updates were performed, the version vectors of the different copies in different partitions diverge, requiring manual conflict resolution.

Established pessimistic approaches are primary site, active replication, and voting. The *primary site* approach supports *k-resilient* data objects, which means that operations on the data can still be performed if up to k nodes in the system fail. This requires that the data is replicated on at least $k+1$ nodes in the system, with one node acting as the *primary* node and the rest as *backups*. In case of a read request, the primary site returns the requested data. In case of an update (write) request, the primary site sends the request to at least k of its backups, and afterwards it performs the operation and returns the result. All backups perform the update operations as received from the primary site. Now if up to k sites fail, the scheme can mask the failure, otherwise, if more than k sites fail nearly simultaneously, it cannot be masked (since the degree of replication is only $k+1$). In case the primary site fails, a new primary site is elected among the backups.

In the *active replication* approach, which is also called *state machine approach*, there is no primary node. Instead, all replicas are active simultaneously, which requires that they remain mutually consistent and maintain the one-copy serializability property [Jal94]. This can be achieved by using atomic broadcasts, for instance. Again, for achieving *k-resiliency*, data has to be replicated on at least $k+1$ nodes. Since any replica can service any request, it is required that all replicas process the same request in the same order and in the same state so that all will reach the same result in order to preserve one-copy-serializability. This requires the satisfaction of the *agreement* property, for which Byzantine agreement protocols can be used, and of the *order* property, which can be achieved by total ordering based on unique identifiers.

Replica control can also be achieved by *voting*, which means that an operation is only performed on replicated data after it has been collectively decided on by replicas through voting. Thus the voting algorithm prevents the concurrent execution of conflicting operations. The advantage of voting-based methods is that – in contrast to the primary site and active replication approach – they not only mask node failures, but also communication failures. Voting mechanisms can further be static (all parameters are fixed), or dynamic (some parameters may be adapted dynamically as failures and recoveries take place in the system). One example for static voting is *weighted voting*, where each copy of a replicated data object is alloted certain votes. For performing a read or write operation, a node has to collect a read quorum of at least r or a write quorum of at least w votes. Here, w has to be greater than half the total number of votes, and $r + w$ greater than total number of votes, in order to prevent the intersection of any two write quorums, or of any read and write quorum, respectively. An optimization and generalization of this approach is *hierarchical voting*. An example for dynamic voting is *dynamic*

reassignment of votes. Here, if nodes fail, the votes to the nodes are reassigned such that the effect of the failed notes is compensated.

A.4.3. Process Resiliency

In distributed applications where different processes are cooperating to perform a task, all processes that are executing on a single node fail and stop if the node itself fails. In case other processes depend on the failing process, this may lead to the failure of the entire distributed computation even if all the required data is available. Therefore, the goal of *process resiliency* is to ensure that a distributed computation proceeds even if some of its constituent processes fail. For separate processes that do not communicate with other processes, simple checkpointing can be used to save consistent states of the process regularly. In the case the process fails on a node, only this process needs to be rolled-back, and it can simply be restarted on some backup node from its last checkpoint. However, when processes communicate and depend on each other, more complex schemes are required to achieve process resiliency.

If a system uses (synchronous) remote procedure calls, where one procedure A calls another procedure B and suspends until the call to B returns, these calls can be made resilient to node failures. This can be achieved by applying the primary site concept, using one primary process and one or more backup processes for processing procedure calls, yielding a procedure similar to the one used for resilient data replication.

Alternatively, the remote calls themselves can be replicated. An example for this is the so-called *circus* approach, where a module offering a certain procedure is replicated on many nodes. The set of module replicas is called a *troupe*. So each module is represented by a troupe, which can act as a server troupe and receive procedure calls from another troupe, or act as a client troupe and make calls to other troupes in return. Whenever a module (or its client troupe) makes a call to a remote module (the server troupe), each member of the client troupe sends its request to each of the procedures of the server troupe. This yields *one-to-many* calls from the perspective of a procedure of a client troupe, and *many-to-one* calls from the viewpoint of the replicated server procedures, and overall results in a *many-to-many* call from client troupe to server troupe. The desired semantics of the many-to-many call are normally that each member of the server troupe executes each request exactly once, and that each member of the client troupe receives all the results (*exactly-once execution*). Depending on the kind of failures that have to be tolerated, a client may just take one of the results (in the case of fail-stop processors), or perform a voting (in the case of Byzantine failures). Either way, in the replicated call approach, the client members have the capability to perform error detection or even error correction.

If processes pass messages among one another in an asynchronous way, resiliency against process failures can be achieved by checkpointing and rolling-back the whole system to earlier consistent states. Alternatively, message logging can be used to recover lost messages, which then requires only the failed process to roll-back to be restarted on a different node using the previously saved checkpoint.

A.5. Fault-Tolerant Software

Software is a totally conceptual entity that has no physical properties. As a consequence, software faults cannot be the cause of physical faults but are always *design faults* [Jal94]. This means that software faults are always the result of incorrect design or erroneous programming ("bugs") caused by human errors in the software. Hence, the goal of *fault-tolerant software* is to cope with design faults in software components. Note that, strictly speaking, fault-tolerant software is different from *software fault-tolerance*, since the latter comprises all techniques for fault tolerance that are supported in software, which also includes those techniques that are designed to handle failures in hardware components. However, in literature, the two terms are often used interchangeably as they are not standardized and used loosely.

Techniques for fault-tolerant software can be classified according to the kind of distributed system they are targeting. This can either be a uniprocess system, where a single process is executing the software that has a design flaw, or a multiprocess system where multiple processes are communicating with each other by means of message passing.

The development of countermeasures against incorrect design and programming of software represents a general challenge of distributed computing and its subdomains. In this dissertation, however, we do not consider software design faults (and thus fault-tolerant software). Instead, we focus on the investigation of dependability challenges that are intrinsic and fundamental to ubiquitous computing in particular.

B. Ubiquitous Computing

In the following sections, we first describe the fundamentals of ubiquitous computing and its related disciplines. Then we give an overview of major technical challenges of ubiquitous computing as they are perceived in the research community today, with the main emphasis on human-computer interaction, context-aware computing, and sensor networks.

B.1. Vision

Mark Weiser described the vision of *ubiquitous computing* as the goal of "enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user" [Wei93b]. He concluded that "the most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it" [Wei91].

Weiser further stated that – unlike virtual reality – "ubiquitous computing endeavors to integrate information displays into the everyday physical world", that it aims at augmenting "the nuances of the real world", envisioning "a world of fully connected devices, with cheap wireless networks everywhere". In contrast to conventional computing systems, ubiquitous computing explicitly aspires to transform the real world in every day life situations, by providing the technical and conceptual means for enabling anytime, anywhere, anyhow computing. This view is also supported by leading IT experts, as former IBM CEO Lou Gerstner indicated during his keynote speech at CeBIT '98 in Hanover, Germany [Ger98]: "The next milestone is what we call 'pervasive computing.' [...] Chips are getting so small and inexpensive, they're being embedded in everything: cars, appliances, tools, doorknobs, clothes. Most significantly, all these tiny intelligent devices will be interwoven in the global fabric of computing and communications."

B.2. Background

In the following, we give an overview of other areas of computing that are related to ubiquitous computing, together with a brief discussion of their fundamental research challenges as they are perceived in the research community today.

B.2.1. Distributed Systems

Distributed algorithms and systems have been the subject of intense development since the 1980s. Generally speaking, distributed (computer) systems qualify all computer applications where several autonomous computers, processors or processes cooperate in some way, as defined by Tel [Tel00], for example. Tel gives a

number of examples where distributed systems are required instead of sequential systems:

- *Information exchange*, e.g., e-mail or World Wide Web;
- *resource sharing*, e.g., printer and file sharing;
- *increased reliability through replication*, e.g., RAID storage systems;
- *increased performance through parallelization*, such as multiprocessor systems, for instance; and
- *simplification of design through specialization*, such as modular design techniques, for example.

Practically, the process of decentralization of computing power and resources was first initiated with the development of the microprocessor, which "burst the shrine" of the prevailing monolithic mainframe systems, and which laid the foundations for the proliferation of personal computers in the office and home environments in combination with local area networks [Sat01, HMNS01]. This process was accompanied by advances in hardware, software, and large-bandwidth communications technologies, such as client/server architectures, universal Web browsers, and distributed databases, which promoted distributed computing. Thus it became possible for parts of a database to be stored and maintained at different locations, for users to take advantage of economical or specialized processing at remote sites, for decision-makers to collaborate across computer networks to make decisions, and for large archives to offer access to their data to anyone connected to the Internet.

Today, distributed systems are no longer limited to local systems, but they become distributed globally, making extensive use of the Internet, an effect which Milenkovic et al. called *Internet Distributed Computing* [MRK+03]. They argue that the Internet itself is in the process of evolving into a "distributed computing platform of unprecedented scale", which includes "Internet uses" based on peer-to-peer and grid computing. And the process of decentralization is expected to continue, which is currently manifesting itself in the appearance of ubiquitous (or pervasive) computing, as Hansmann et al. explain [HMNS01]: The computer is "irresistible on its way to push all limits and is getting omnipresent", eventually becoming a "part of everyday life and an inevitable component when performing a variety of private and business related tasks". Beyond the era of personal computing, ubiquitous computing makes "information access and processing easily available for everyone from everywhere at any time", enabling users to "exchange and retrieve information they need quickly, efficiently, and effortlessly, regardless of their physical location".

As Satyanarayanan [Sat01] points out, this ongoing progress of ubiquitous computing systems largely benefits from many areas of distributed systems which are "foundational to pervasive computing". According to Satyanarayanan, this is in particular the case with respect to the classic distributed systems research fields of *remote communication* (e.g., protocol layering and remote procedure calls), *fault tolerance* (such as atomic transactions and two-phase commit protocols), *high availability* (including optimistic/pessimistic recovery and replication strategies), *remote information access* (such as caching, distributed file systems, and distributed data bases), and *security* (such as mutual authentication based on encryption). Future distributed system challenges include the support of mobile code, multimedia data

streams, user and device mobility, and spontaneous networking [CDK00]. Scalability, quality of service, and robustness issues with respect to partial component failures are expected to become key issues. In the process, many emerging essential techniques will be incorporated into the area of ubiquitous computing [MS03].

B.2.2. Embedded Computing

The forthcoming realization of the ubiquitous computing vision asserts itself in the continuing progress in the development of small and cheap computing and communication technologies [HMNS01], which largely benefits from the technological advances in the domain of *embedded computing* [Mat05]. As the power of microprocessors, storage capacities and communication bandwidth rapidly increase, it becomes technically feasible to build ever smaller, cheaper and more abundant computers. This development ultimately results in the creation of so called "smart things" which not only have access to the Internet and its plentiful resources, but which also are increasingly capable of autonomous cooperation and interaction with each other [Mat03].

Embedded computing systems can be defined as special-purpose computer systems which are completely encapsulated by the devices they control. An embedded system usually has specific requirements and performs pre-defined tasks, unlike a general-purpose personal computer. Important examples where embedded systems are employed are sensor networks, fly-by-wire systems, engine control (e.g., for improved fuel efficiency and lower emissions in automobiles), medical implants and monitoring systems, avionics (e.g., navigation and collision avoidance), smart homes and work spaces, and space control. But there are also more mundane examples of how embedded systems pervade our everyday life environment, spanning the whole spectrum of electronic devices and products, such as electronic toys, multimedia entertainment systems, mobile (smart) phones, digital alarm clocks, and even toasters and coffee machines with built-in or embedded computer systems.

Research Challenges. The following are characteristic areas of research and development as they are perceived today in the research community of embedded computing [Ano01, Tec04, JEC05]:

- *Embedded hardware support:* The core of an embedded system is the used hardware platform and technologies. On the one hand, this comprises the development of miniature, low-power digital signal processors (DSPs), microprocessors, and systems-on-a-chip. On the other hand, it is related to research in the area of hardware specification, synthesis, modeling, simulation, analysis, including the investigation of aspects of power-awareness, reliability and fault-tolerance, security, functional verifiability, and performance modeling.

- *Embedded software:* Embedded hardware cannot be operated without suitable embedded software that takes full advantage of the provided hardware capabilities, and which defines the program tasks and services to be executed on the hardware platform. The development of embedded software comprises the study of compilers, assemblers and cross assemblers for programming, including aspects of memory management, object-oriented pro-

gramming, virtual machines, scheduling, concurrent software, distributed and resource aware operating systems, and middleware support.

- *Embedded system architecture:* Apart from basic hardware and software support, advanced concepts are required for building more complex embedded system architectures. Related challenges include the investigation of heterogeneous multiprocessor systems and reconfigurable platforms, as well as communication issues, protocols, on-chip network capabilities, and embedded microcontrollers.

- *Hardware/software co-design:* The particular properties of embedded systems (such as miniature size, low power, and resource constraints) demand special tools and methodologies supporting the design and development process. This includes the development of tools for testing and debugging, specification and modeling, and design representation. It also requires the study of the interaction and interrelationship between architecture and software design, of algorithms, design concepts, software synthesis and re-use, and a theoretical analysis and exploration of the available design space and its boundaries.

- *Real-time systems:* Embedded systems often have to meet great demands on response time and real-time operability. Therefore, an important research field is the investigation of real-time related aspects such as software, distributed real-time systems, real-time kernels, real-time operating systems, multitasking, and task scheduling.

- *Testing and verification:* Due to size and hardware restrictions, the testing and verification of the design and proper functioning of embedded computing platforms is a challenge. This includes issues related to the design-for-test, design verification, test synthesis, built-in self-test, embedded testing, and verification for embedded and system-on-a-chip systems.

- *Application-specific processors and devices:* Embedded computing platforms can be tailored for performing tasks with high performance and efficiency. Typical areas for dedicated, application-specific processors and devices are network processors, real-time processors, media and signal processors, application specific hardware accelerators, reconfigurable processors, low power embedded processors, bio/fluidic processors, and communication processors (such as for Bluetooth and ZigBee). It also includes complete products such as handheld devices, for instance.

- *Wireless ad hoc communication and networking:* Prominent research issues are theoretical boundaries (throughput, optimal routing strategies, multihop-protocols), technical challenges such as robustness and quality-of-service guarantees, efficient low-level protocols and hardware technologies for efficient single-hop wireless ad hoc communication.

- *Sensor networks:* Embedded computing technology and platforms provide the hardware and software backbone of sensor networks. As embedded computing moves from working in closed, isolated systems towards communicating, networked, distributed solutions and open application platforms, research

challenges also comprise the domains of wireless and ad hoc networking technologies, methods of self-organization, architectures, protocols, software, and applications. Recently sensor networks have attracted considerable attention in research and industry alike and become an independent research domain. For an overview of research issues and challenges related to sensor networks, please refer to Section B.6.

- *Dependability:* Embedded systems are threatened by enormous security issues which challenge their technical and economical viability. This calls for security technologies, concepts and architectures with respect to software and hardware. A particular concern is the rapidly growing complexity of embedded systems, which on the one hand is caused by the enormous technological advances such as in the field of nanotechnology, and on the other hand by a significant increase in heterogeneity: there is an ever growing variety of transducer devices, sensor technologies, actuators, interactive screens and displays, input and output devices (speech, handwriting, implicit input based on context awareness and activity recognition), computing devices and smart appliances, and communication issues (standards, protocols, radio frequency). Further dependability challenges are complexity management, fault containment, and control of dynamic and highly distributed embedded systems [Rus01b, Rus01a, Kop04], the delivery of quality-of-service guarantees, reliable real-time communications, and methods for energy preservation [RV03] and harvesting. A particular challenge is the development of safe embedded software [Lev02], as the complexity of most embedded software limits the ability to assure safety and to prevent hazardous behavior.

- *Emerging new topics and challenges:* New challenges for next generation embedded computing systems arise from new technologies, such as nano- and biotechnology, new engineering principles, and new application domains, such as embedded Internet tools and ubiquitous computing in general.

B.2.3. Mobile Computing

Mobile computing can be considered as the logical forerunner of ubiquitous computing. Since motion is an integral part of everyday life, ubiquitous computing technology must support mobility. If this is not the case, a user will be acutely aware of the technology by its absence when he moves [Sat01]. Back in 1994, Forman and Zahorjan described an impending paradigm shift from using self-contained and statically networked computers to the employment of portable computers capable of wireless networking [FZ94]. They referred to this emerging technology as *mobile computing*, identifying its most distinguished feature as being the "elimination of time-and-place restrictions", granting the user "access to digital resources at any time, from any location". More generally speaking, computing can be considered to be mobile if it can occur in moving locations, such as in vehicles, on aircraft, or carried on the body. Due to the mobility aspect it is impracticable to use wired communication among mobile entities, making wireless communication particularly important for mobile computing. Ubiquitous computing, in comparison, takes this development even further by transforming the user's environment as a whole into a conglomeration of smart computerized objects and devices, which

not only aggravates the problems of mobile computing, but which also introduces a number of additional challenges and research issues, as we explain in the following sections.

Research Challenges. Talking about mobile computing challenges, Forman and Zahorjan [FZ94] stated that the expected increase in utility, versatility and flexibility of future mobile computing systems brings about its own particular problems and challenges, stemming from "three essential properties of mobile computing: communication, mobility, and portability." Especially the mobility aspect introduces four novel key constraints in comparison to traditional distributed computing systems: an unpredictable variation in network quality, lowered trust and robustness of mobile elements, limitations on local resources imposed by weight and size constraints, and concern for battery power consumption [Sat96].

Because of the strong need for wireless connectivity in mobile computing, *mobile networking* was early recognized as an important issue, including the study of reliable and efficient ad hoc protocols and high-performance techniques [Sat01]. In general, wireless communication is characterized by "lower bandwidths, higher error rates, and more frequent spurious disconnections", which can also lead to an increase in communication latency [FZ94]. Hence important challenges related to wireless communication are low bandwidth, high variability of the available bandwidth, heterogeneity of networks and protocols, and the (in)security of wireless links.

There are further technical challenges arising from *mobility*, as described by Satyanarayanan [Sat01], for example. Firstly, there is the issue of *mobile information access*, which includes the areas of disconnected operation, bandwidth-aware file access, and selective control of data consistency. Secondly, mobility-induced changes in the availability of resources enforce the need for *adaption* [Sat96], providing support for adaptive applications by means of proxy-based transcoding and adaptive resource management, for instance. Thirdly, due to an increased dependence of mobile computers on location-specific context information, *location sensitivity* is another important area. Here the focus lies on location sensing and location-aware system behavior, which recently has become the explicit target of *location-aware computing* research. Mobile devices may require location-dependent information for the dynamic reconfiguration of their services according to the particular location-dependent resources and services whose availability may again be static or dynamically changing [FZ94], calling for location-aware services and service configuration.

A further mobility-induced challenge is address migration for mobile devices that frequently change their physical location and, in the process, also change their network address while roaming between different network domains. Nevertheless, in the last decade there has been significant effort put into the development of networking protocols with special support for device mobility, such as mobile-IP [Per98] and enhancements of the IPv6 protocol [QLIM01].

Forman and Zahorjan further consider "migrating locality" an issue, arguing that physical distance does not necessarily reflect network distance, so that a small actual movement could lead to a disproportionate growth of the communication path, which could potentially result in longer latency and a higher risk of disconnections "when crossing network administrative boundaries." However, the latter has not

proven to become a real concern today, as longer communication paths within today's evolved high performance networks are not significantly affecting latency.

The third key objective of mobile computing according to Forman and Zahorjan is to achieve a high degree of *portability* of hand-held mobile computers. This leads to further challenges that have their origin in the portability constraints imposed by size and resource limitations. The most prominent constraint is low power, as batteries typically constitute the largest single source of weight in a portable computer. This remains a crucial issue today, especially since the development rate of more powerful and more compact batteries is still far behind the development rate of computer hardware and circuits. To alleviate the problem, power consumption should be minimized in the first place, either by designing power-aware computer components and system-level energy saving techniques [Sat01] (e.g., adaptive power saving mechanisms that allow to turn off idle system components or to temporarily reduce the clock speed of the processor), or by developing power-efficient applications.

Another important issue in the context of portability is the increased *risk* of *physical damage*, *unauthorized access*, *loss*, and *theft*, which may result in breaches of privacy or total loss of data. As a solution, Forman and Zahorjan suggested to minimize the essential data kept on the portable device, to use encryption, and to employ data synchronization and replication mechanisms. Further challenges are small user interfaces and small available storage capacities, both of which are direct results of the size constraints typical of portable devices.

B.2.4. Peer-to-Peer Computing

Peer-to-peer (P2P) *computing* constitutes a more recent domain that shows some overlap with ubiquitous computing research. P2P computing systems are generally characterized by the direct sharing of computer resources (such as content, CPU cycles, storage and bandwidth) in a decentralized manner [ATS04]. More precisely, peer-to-peer (P2P) computing can be defined as a "network-based computing model for applications where computers share resources via direct exchanges between the participating computers" [Bar02]. One finds a considerable number of further definitions of "peer-to-peer" in literature, which are mainly distinguished by the "broadness" they attach to the term. However, there is a general understanding about the "fundamental actions that P2P enables users to do", which are the *sharing of resources*, and *collaboration* [Bar02]. Resources that can be shared generally are files (e.g., music, video, etc.), compute-cycles (such as it is the case in the SETI@home [SET05] application), services (e.g., messaging and security), storage space, and information. P2P-based collaboration is an innovative, evolving area, including direct exchange between peers, peer-intermediated transactions, granting access to resources (as in SETI@home), and any kind of interaction that is formed through policies in P2P-based online communities.

P2P-based systems were initially built to implement file-sharing systems that spanned Internet-like environments, with a focus on searching and routing aspects, and the integration of hash-map-like functionalities (mapping keys to locations) [DM04] in distributed settings. P2P computing systems generally abstract from the underlying hardware, assuming a logical model with interconnected processes as logical nodes. These logical nodes are able to self-organize into network

topologies, capable of accommodating transient populations of nodes while maintaining acceptable connectivity and performance without requiring the intermediation or support of a global centralized server or authority. Regularly recurring connectivity and availability of the distributed physical nodes, which are linked by a communication network, is usually taken for granted. In contrast, ubiquitous computing systems explicitly change or extend the quality of physical objects [Mat03] to provide novel services based on the collaborative effort of these augmented smart objects. In doing so, smart objects often collaborate in an ad hoc fashion, communicating via short-range ad hoc wireless links rather than via infrastructure-based communication networks.

Peer-to-peer architectures have been employed for a variety of different application categories, which include the following classes of P2P applications [ATS04]:

- *Communication and collaboration.* This category includes systems that provide the infrastructure for facilitating direct, usually real-time, communication and collaboration between peer computers. Examples include chat and instant messaging applications.

- *Internet service support.* A number of different applications based on peer-to-peer infrastructures have emerged for supporting a variety of Internet services. Examples of such applications include peer-to-peer multicast systems.

- *Distributed computation.* This category includes systems whose aim is to take advantage of the available peer computer processing power (CPU cycles). This is achieved by breaking down a computing-intensive task into small work units and distributing them to different peer computers that execute their respective work units and return the results. Central coordination is invariably required.

- *Database systems.* Considerable work has been done on designing distributed database systems based on peer-to-peer infrastructures.

- *Content distribution.* Most of the current peer-to-peer systems fall within the category of content distribution, which includes systems and infrastructures designed for the sharing of digital media and other data among users.

Research Challenges. *Decentralization* is a key attribute of P2P, which consequently puts *communication* and *connectivity* at the heart of P2P computing. The development of platform-spanning communication protocols and standards suitable for P2P applications is a fundamental requirement. Here, a particular challenge is the consideration of firewalls and existing Network Address Translation (NAT) technologies, which restrict or hamper direct access to the internal systems and resources of organizations [Bar02].

The distributed sharing of resources and the dynamics of the P2P environment raises further fundamental issues of P2P computing. Once communication is ensured between peers, the next challenge to be addressed is how these resources can be unambiguously named and efficiently searched, found, and delivered, considering that nodes, users, and resources come and go. This requires the study of adaptive *discovery* mechanisms for exploring and detecting available resources in a dynamically changing environment, *directory* services that manage information

about discovered resources, and *search* mechanisms that allow users to efficiently locate the particular resources they are interested in [Bar02]. Further, since resources such as files may change their locations frequently or be replicated across multiple peers, there is a need for P2P naming schemes that provide names that are independent from physical locations, and supporting *content*-based references and discovery procedures.

An open issue is *interoperability* between P2P systems. For most established P2P systems, interoperability is not (yet) an issue. However, in the long run, interoperability would strongly facility the migration or development of new P2P applications, because these systems could directly rely on the available basic services [Bar02]. As a result, development time could be reduced, and the potential user community for new applications would be bigger from the start, increasing the chance that the provided P2P services will actually establish themselves and be used by a larger community.

A key problem of P2P systems is the lack of cooperation (free riding) among participating nodes. A paramount challenge is the development of incentive techniques for peer-to-peer networks, which is a difficult task considering the large populations, high turnover rate of nodes, the symmetry of interest, zero-cost identities, and the possibility of collusion among nodes or "traitors" [FLSC04].

Since P2P computing provides mechanisms for direct and content-rich exchanges between users, P2P can also be considered a tool for creating online communities [Bar02]. Since P2P communities are dynamic (individual members may choose to join and leave at any time), these communities have to be *self-organizing*, enforcing certain policies and rules of conduct even in the absence of a central authority. Furthermore, the members of a community usually have a shared common interest, which determines the nature of the exchanges, shared resources, and the access granted to those resources. In such a setting, each peer should be able to control the access permissions granted to other peers. So an important research challenge is the development of concepts for the formation and management of users and communities that enable a strong *local autonomy* of the user. Since it cannot be expected that peers are connected all the time, P2P applications need to be particularly *resilient* and *fault-tolerant* to deal with the intermittent presence of peers. More generally, peer-to-peer architectures require the "ability to treat instability and variable connectivity as the norm, automatically adapting to failures in both network connections and computers, as well as to a transient population of nodes" [ATS04]. This "fault-tolerant, self-organizing capacity" calls for an "adaptive network topology that will change as nodes enter or leave and network connections fail or recover, in order to maintain its connectivity and performance". Mechanisms are therefore required that ensure a high availability of resources, which can be achieved by means of data and process replication across peers, for example. This also requires fault-tolerant routing and information retrieval mechanisms [ADS02]. A further challenge is to ensure that critical content remains available during network failures, and that the content can be accessed with sufficient performance, which can be achieved by concepts that move content closer to where it is being used, for instance. Furthermore, with potentially many copies of the same content distributed across different peers of the P2P network, ensuring and maintaining content coherency and synchronization is a non-trivial challenge. This problem is aggravated by temporary losses of network connectivity, and by the temporary or

permanent unavailability of one or more peers.

Among the highest ranked concerns with regard to P2P computing, however, are counted *security* [CDG+02] and lack of *trust* [Bar02]. Peers are often connected for long periods of time, offering resources and services to other, often unknown, untrusted and potentially malicious peers. Basic security requirements are peer authentication (to know and control who is accessing a system), authorization (to control what a peer is allowed to do on a system), and data integrity (protection against tampering with data).

As P2P communities grow in numbers of peers, the amount of offered resources and the number of requests for accessing those resources increases accordingly. This causes challenges with regard to performance and scalability. The performance of a P2P system to a large extent depends on the available network bandwidth, and latency. The latter is of particular importance with regard to real-time applications, such as instant messaging, or online gaming. Scalability may become an issue with regard to central or distributed naming schemes, and with directory services, including searches and resource discovery [Bar02].

B.2.5. Grid Computing

A not so closely related modern branch of distributed computing research is called *grid computing*. Computational grids are distributed systems that enable the large-scale coordinated use and sharing of geographically distributed resources, constituting "ensembles of distributed, heterogeneous resources" [Cas02]. Computational grids have emerged as popular platforms for deploying large-scale and resource-intensive applications, and currently large collaborative efforts are undertaken to provide the necessary software infrastructure. In contrast to highly dynamic and diverse ubiquitous computing systems, grids are based on persistent, standards-based service infrastructures, often with a high-performance orientation [FKT01]. However, there are ongoing efforts of integrating the mobile wireless computing world with computational grids [PHD02].

The technical foundations of grid computing are found in the tremendous technological progress, leading to a nearly annual doubling of data storage capacity and network performance (relative to computer speed). At the same time, computer power "only" doubles at a rate of approx. every 18 months, and thus is "falling behind storage" [Fos02]. Foster argues that, if networks outpace computers at this expected rate, communication becomes essentially free. As a consequence, it suggests itself to "exploit this bandwidth bounty" by imagining "new ways of working that are communication intensive, such as pooling computational resources, streaming large amounts of data from databases or instruments to remote computers, linking sensors with each other and with computers and archives, and connecting people, computing, and storage in collaborative environments that avoid the need for costly travel".

But as Foster points out, the Grid goes beyond sharing and distributing data and computing resources. If offers scientists new and more powerful ways of working. According to Foster, examples of this are:

- *Science portals*, making advanced problem-solving methods easier to use. For example, sophisticated packages can be invoked remotely from Web browsers or other simple, easily downloaded "thin clients", or they can be run remotely

on suitable computers within a Grid. Such science portals are currently being developed in biology and computational chemistry, for instance [Fos02].

- *Distributed computing*, by employing high-speed workstations and networks as a means to yoke together PCs within an organization to form substantial computational resources.

- *Large-scale data analysis*, by harnessing distributed computing and storage resources. An advantage hereof is that the natural parallelism inherent in many data analysis procedures makes it feasible to use distributed resources efficiently. Further, for various technical and political reasons, assembling these resources at a single location often appears impractical, whereas the collective institutional and national resources of a bigger number of institutions participating in large-scale experiments can provide these resources. These communities can then not only share computers and storage, but also analysis procedures and computational results.

- *Computer-in-the-loop instrumentation*, enabling a quasi-real-time analysis which greatly enhances an instrument's capabilities. Many scientific instruments such as telescopes, synchrotrons, and electron microscopes that generate raw data streams would benefit from such a real-time analysis: it makes it possible to respond to observed phenomena while they are still active, which is not possible if the raw data is archived for subsequent a posteriori batch processing. An example for this is the ability to zoom in on active solar flares that are brief and sporadic while they occur.

- *Collaborative work*, enabling the aggregation of human expertise along with data and computing power. This would allow collaborative problem formulation and data analysis, which is important to Grid applications, by researchers who are working in different, possibly remote physical locations.

General Research Challenges. Foster described *authentication*, *authorization*, and *policies* to be among the most challenging issues in Grids [Fos02]. He argues that while traditional security technologies are concerned primarily with securing the interactions between clients and servers, the situation is more complex in Grid environments, where the distinction between client and server tends to disappear. In Grids, an individual resource can act as a server one moment (as it receives a request) and as a client at another (as it issues requests to other resources). This leads to a number of interesting requirements of Grid systems, such as:

- *Single sign-on*, allowing a user to authenticate him or herself only once and then assign to a computation the right to operate on his or her behalf for a specific period, rather than requiring him or her to re-authenticate on each occasion a single computation needs to access other distributed resources. One way for achieving this capability is the creation of a so-called proxy credential.

- *Mapping to local security mechanisms*, enabling the integration of different local security solutions found at different sites, such as Kerberos and Unix. By mapping to these local solutions at each site, local operations can proceed with appropriate privileges.

- *Delegation*, enabling computations that span many resources, and which require the creation of subcomputations (or subsidiary computations) that may themselves generate requests to other resources and services, possibly creating additional subcomputations, and so on. Authentication operations – and hence further delegated credentials – are involved at each delegation step, as resources determine whether to grant requests and computations determine whether resources are trustworthy. An important challenge is to limit the risk that credentials are acquired and misused by an adversary, which is the greater the further these delegated credentials are disseminated.

- *Community authorization and policy*, since resources (and users) need to be able to express policies in terms of other criteria, such as group membership, as it is infeasible for each individual resource to keep track of community membership and privileges. One possible solution is to identify resources with a cryptographic credential issued by a trusted third party together with a community authorization system that allows the delegation of policy decisions to a community representative.

Concerning the importance and impact of Grid computing in the future, Foster claims that "it will surely take longer than some expect before Grid concepts and technologies transform the practice of science, engineering, and business, but the combination of exponential technology trends and R&D advances [...] are real and will ultimately have dramatic impacts". He further predicts that "in a future in which computing, storage, and software are no longer objects that we possess, but utilities to which we subscribe, the most successful scientific communities are likely to be those that succeed in assembling and making effective use of appropriate Grid infrastructures and thus accelerating the development and adoption of new problem solving-methods within their discipline".

An underlying analogy describing the vision of Grid computing was given as early as 1965, likening the Grid to the electrical power grid [Fos02], providing access to computation and data in an "easy, pervasive, standard way as plugging in an appliance into an outlet" [Cas02]. Here it shows that with the underlying goal of providing a pervasive, ubiquitous infrastructure, it is reasonable to suppose that grid computing offers some definite points of contact with the domain of ubiquitous computing. This expected potential for synergies at the interface between ubiquitous computing and grid computing has already moved into the focus of research. One idea, for instance, is to integrate mobile wireless devices into the computational grid. Phan et al. describe the challenge of "harvesting the increasingly widespread availability of Internet connected wireless mobile devices such as PDAs and laptops to be beneficially used within the emerging national and global computational grid" [PHD02]. They argue that, due to inherent resource limitation, the integration of mobile wireless consumer devices into the Grid initially seems unlikely and unprofitable. But in the long run, due to their potentially enormous numbers, mobile devices form an untapped abundance that could be successfully integrated into grids, by means of a proxy based, clustered system architecture, for example.

Vice versa, assuming the availability of a ubiquitous, easy to tap Grid infrastructure, Grid computing could provide valuable resources and services to resource-limited mobile devices and smart objects in ubiquitous computing environments,

enabling the development of more efficient, powerful, and dependable applications. However, this topic is beyond the scope of this dissertation and should be addressed as part of future ubiquitous computing research.

B.3. Ubiquitous Computing Technologies

The continuing advances in microelectronics and embedded computing lead to a degree of miniaturization of computer technology that enables the development and production of *microprocessors*, *memory*, *wireless communication* technologies, *sensors*, and *energy sources* of ever decreasing form factors and weight [Mat05]. Together with novel *input/output devices*, these technologies can be considered the enabling technologies for the realization of ubiquitous computing systems and applications.

Owing to their miniature size, their low energy consumption and their low price, microprocessors, memory modules, communication chips, and sensors can be *embedded* into a broad range of everyday life artifacts. This way it has become feasible to even render inanimate, traditionally non-electronic things into "smart" computerized objects capable of executing certain services and applications. Thanks to their added communication capabilities, *smart objects* are able to interact and cooperate with other smart objects, either by means of ad hoc communication with physically proximate objects residing in the immediate vicinity, or by infrastructure-based communication with remote objects that are connected to a (local) Wireless LAN or to the (global) Internet. Ultimately, such smart cooperating objects are expected to form the basis for the emergence of an Internet of things [Mat05].

B.3.1. Wireless Communication

Interaction and cooperation between distributed computing devices requires communication among those devices. Further, portable, mobile devices that are not connected by wire to a communications network while being stationary or in transit need some means of wireless communication to be able to interact with other (mobile or immobile) devices.

The two most popular and widespread mediums for wireless communication between computing devices are *infrared* (IR) light and *radio frequency* (RF).

IR-based communication can be considered the forerunner of standardized wireless communication technology. IR technologies are particularly suited for short distance wireless communication channels with low-to-medium data throughput [Bak04]. IR communication systems are simple to build and design, low-priced, and provide a stable connection without creating possibly harmful interference. This enables their deployment in areas sensitive to RF interference such as hospitals and aircraft [NTT03]. Directed IR communication can further be of particular interest for the use with mobile, portable devices, since it provides a "point & shoot" capability.

Using IR communication typically allows transmission rates up to 4 Megabits/s, with envisioned future data rates in the range of up to 100 Megabit/s [IDA05]. However, in everyday computing equipment such as laptops or PDAs, IR hardware with a maximum transmission rate of only 115.2 kbit/s is most commonly used [Bak04]. A disadvantage of IR communication in comparison to RF communication is that it only supports point-to-point communication. Further, for

achieving the potential higher data rates with acceptable error rates, the IR sender and receiver have to be aligned to enable direct line of sight communication.

With the development of RF-based wireless communication technologies the alignment and line of sight restrictions of IR communication were overcome. At the same time, significantly higher data rates became possible, reaching the order of magnitude of Gigabit/s. The breakthrough of wireless RF-based communication was achieved with the introduction of the IEEE 802.11 family of wireless networking standards, which are commonly referred to as Wireless LAN (WLAN) or Wi-Fi (short for Wireless Fidelity) technology [WFA06]. Wi-Fi networks based on the IEEE 802.11b standard, featuring data rates up to 11 Mbit/s, have become commonplace in public places and spaces. The further development of wireless communication standards resulted in even higher data rates of 54 Mbit/s (IEEE 802.11g) and beyond. The appearance of portable, mobile devices created a demand for energy-efficient short range communication technologies with particular suitability for wireless ad hoc communication. This led to the development of specialized short-range wireless communication technologies and protocols, such as Bluetooth [BS06], and ZigBee (IEEE 802.15.4) [Zig06], for example.

B.3.2. Embedded Computing Technologies

The technologies that enable embedded computing comprise the whole spectrum of computer hardware, with a strong focus on miniaturization and power-efficiency. In the following, we give a brief overview of typical embedded computing technologies, which are usually employed in ubiquitous computing systems:

- *Microprocessors*, such as digital signal processors (DSPs), systems-on-a-chip, generic and customized microprocessors based on very/ultra large scale integration (VLSI/ULSI) architectures, and field programmable gate arrays (FPGAs).

- *Small-scale data storages*, such as highly integrated memory chips, standalone memory storage devices such as compact-flash cards, secure digital cards, microdrives, and USB memory sticks.

- *Highly integrated circuits for ad hoc communication, networking, and interfacing*, such as wireless RF communication chips supporting Wi-Fi (IEEE 802.11x), Bluetooth, and ZigBee; communication interfaces and bus architectures, such as serial and parallel ports, Universal Serial Bus (USB), Firewire, and the Inter-IC (I2C) multi-master bus.

- *Displays*, such as light emitting diodes (LEDs), thin film transistor (TFT) displays, and more recently also flexible organic [Kid99] or polymer [Hel00] displays.

- *Sensors*, such as miniature video cameras, microphones, accelerometers, gyroscopes, pressure sensors, and sensors for temperature, humidity, light intensity, and electrostatic capacity.

- *Actuators*, such as miniature switches and controls, and communication-based remote notification, logging, messaging, and auditing mechanisms.

- *Sensor nodes*, i.e., autonomously operating small-scale sensor platforms, such as BTnodes [BKM+04] and the Mica wireless platforms (also known as Berkely Motes) [HC24], for instance.

This list is not and cannot be exhaustive, since embedded computing technologies and their potential application in the context of ubiquitous computing constitute an actively evolving field. It is expected that, in the long run, the realization of various technologies that today are still being considered utopistic or exotic will become practical and feasible, such as holographic storage media, or Gigabit wireless communication technologies, for instance [Mat05].

B.3.3. Smart Everyday Objects

Everyday life objects and artifacts that are augmented with embedded computing capabilities have gained considerable importance in ubiquitous computing scenarios. The process of augmentation changes or extends the quality of the original objects, such as their mode of functioning, usage, capabilities, etc., thus rendering them "smart" [Mat03] in some way. Siegemund [SFV04] gives a concrete definition for such *smart everyday objects* (in short: *smart objects*), which he defines as everyday artifacts augmented with small sensor-based computing platforms. Such smart objects can perceive their surroundings through sensors, and collaborate with peers using short-range wireless communication technologies, which makes them *aware* of their environment. They further provide context-aware services to users or other smart objects in ubiquitous computing environments.

According to Siegemund, the computational capabilities of smart objects are usually very limited, because their embedded computing platforms have to be small and unobtrusive. And since smart objects often do not possess conventional I/O interfaces such as keyboards or displays, their means of interaction with users are very restricted. Furthermore, due to their limited energy resources, smart objects mainly support short-range communication technologies only.

Siegemund showed that some challenges imposed by these resource restrictions can be overcome when nearby smart objects are able to spontaneously cooperate, and to access the capabilities of nearby handheld devices. This is especially true for ubiquitous computing environments where people are expected to carry personal devices and smart objects with them. Siegemund describes an architecture that allows the exploitation of the features of nearby handheld devices in an ad hoc fashion [SFV04]. As a result, he identified the following means by which computer-augmented everyday artifacts can make use of handhelds: (1) as mobile infrastructure access point, (2) as user interface, (3) as remote sensor, (4) as mobile storage medium, (5) as remote resource provider, and (6) as weak user identifier. Siegemund further developed an architecture that supports the local cooperation of smart objects based on distributed tuple spaces [Sie04a].

This view of super distributed interacting objects is congruent with the idea of interconnected smart objects of ubiquitous computing, such as described by Weiser and Brown [WB97], for instance, stating that "Ubiquitous Computing is fundamentally characterized by the connection of everyday things in the real world with computation".

The Object Management Group (OMG) [OMG05] is in the process of proposing the standardization of smart objects, for which they use the term *super distributed*

object (SDO) [OMG04]. In contrast to smart everyday objects, which explicitly refer to physical entities that locally interact and share data, the SDO approach aims at a more general view on smart objects: an SDO provides an abstraction for both software and hardware entities with a focus on the flexible ad hoc composition and/or configuration of services. Quoting from the "Platform Independent Model (PIM) & Platform Specific Model (PSM) for Super Distributed Objects (SDO)":

> A Super Distributed Object (SDO) is a logical representation of a hardware device or a software component that provides well-known functionality and services. One of the key characteristics in super distribution is to incorporate a massive number of objects, each of which performs its own task autonomously or cooperatively with other objects. Examples of SDOs include abstractions of devices such as mobile phones, PDAs, and home appliances, but are not limited to device abstractions. An SDO may abstract software component and act as a peer in a peer-to-peer networking system. SDOs provide various different functionalities (e.g., TV set, refrigerator, and light switch) and abstract underlying heterogeneous technologies. They are organized in an ad hoc manner to provide an application service in mobile environments [OMG01]. For other characteristics in super distribution, please refer the Super Distributed Objects Whitepaper [OMG01].
>
> Today, there are several resource interconnection technologies such as Universal Plug and Play, HAVi, OSGi, ECHONET, and Jini. They are, however, restricted to specific platforms, network protocols, and programming languages or they focus on limited application domains. No common model-based standards exist to handle various resources in a unified manner independently of underlying technologies and application domains. The objectives of this specification are to abstract the existing resource interconnection technologies into a higher layer, define their information, and computational models in the layer, and make objects defined in the models interoperable. This specification does not address access control or security aspects.

The use of SDOs has been proposed for the ad hoc creation or adaptation of so called I-centric services based on personal user preferences [FKSK02, ASRPZ03, PZSA04], for instance.

B.3.4. Object Identification Technologies

As a rule, the coordination of the interaction of highly distributed smart entities requires some means of discrimination. More precisely, it is vital to be in a position to uniquely identify single entities, in order to be able to unambiguously relate responsibilities, functionalities, and data to entities. For instance, networked devices require a unique address identifier to enable point-to-point communication, and mobile phones have globally unique phone numbers that enable other mobile phones to contact individual devices among millions of others.

In ubiquitous computing environments, we find the following means of object identification:

- *Unique address identifiers used for networking and in communication protocols.* Examples are logical IP network addresses, and physical MAC addresses of hardware communication interfaces, such as found in networked systems using Ethernet, Wireless LAN, or Bluetooth communication technologies. In order to retrieve the address identifier of an object, the inquiring object has to have the same communications interface at its disposal, and it has to be possible to establish a communication path between the two entities, either directly via ad hoc communication or indirectly via network connectivity. The usage of address identifiers is therefore only an option in settings where both the inquiring and the inquired entities are equipped with corresponding communications technology and connectivity.

- *Visual Codes.* Visual codes (or synonymously: visual tags) are attached to physical objects and generally require line of sight for reading. Prominent examples of such visual codes are one-dimensional barcodes (read by means of an infrared scanner), or two-dimensional codes, such as the Sony CyberCode [RA00] or the VisualCodes [Roh04] developed at ETH Zurich, which can be scanned and detected by means of low-cost off-the-shelf video cameras. The tags themselves usually consist of black and white patterns that encode the ID and sometimes additional meta information that allows the reading device to determine the alignment and distance of the tag with respect to the reader. Typical tag patterns are stripes in the one-dimensional case (barcodes), and two-dimensional patterns of black and white squares in the case of visual codes. The advantage of visual codes over address identifiers is that the first require no computing or communication capabilities on behalf of the visually tagged objects, apart from the tag itself. On the other hand, the scanning of visual codes requires special hardware on behalf of the entity performing the detection.

- *Radio Frequency Identification (RFID) tags.* Radio frequency identification technology has become a replacement for barcodes in many industrial settings. RFID tags are also attached to physical objects that do not need to have computing and communication capabilities of their own. The advantage of RFID tags over visual tags is that they require no line of sight, and no particular orientation of the physical tag with respect to the detector. RFID tags can also be detected over larger distances (up to several meters). Further, if collision detection and resolution is supported by the RFID hardware, multiple tags can be detected in one sweep.

  There are two categories of RFID tags: active and passive tags. Active RFID tags have their own power source, typically a battery, which enables them to actively listen and respond to scanning requests. When enhanced with sensory capabilities, active tags can be transformed into simple sensor nodes that allow to monitor and transmit additional information along with their respective tag IDs. For example, it is thus possible to keep track of physical properties (e.g., temperature and humidity) of perishable goods pallets tagged with enhanced active RFID tags (such as tags of the Identec IQ series [IDE05], for instance).

  Passive RFID tags, in contrast, possess no active power sources of their own.

Instead, when inside the range of an RFID antenna during tag detection, a passive RFID tag draws and uses energy from the antenna field in order to send back its tag ID according to a well-defined signaling and transmission protocol, which may further include mechanisms for collision arbitration and resolution. For an in-depth discussion of RFID technologies and applications, refer to Finkenzeller [Fin03], for instance.

B.4. Human-Computer Interaction

B.4.1. Definition

Human-computer interaction (HCI) is "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them" [HBC+96]. An overview of fundamental topics related to the design and analysis of human-computer interaction systems is given in Fig. B.1.

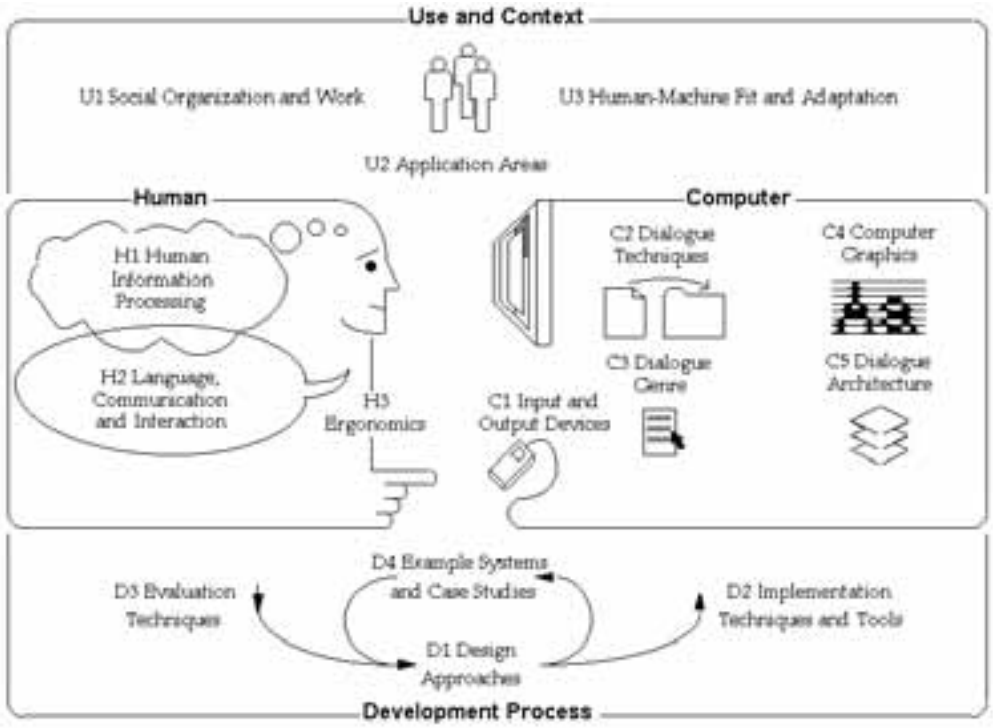

Figure B.1.: Overview of topics in human-computer interaction (Source: [HBC+96])

Hewett et al. state that viable human interfaces are more technology-sensitive than many parts of computer science, especially because human-computer interaction involves "transducers" between humans and machines, and because humans are sensitive to response times [HBC+96]. They also observe that human-computer interaction is particularly affected by "the forces shaping the nature of future computing", among which they count the following trends:

- Decreasing hardware costs leading to larger memories and faster systems.
- Miniaturization of hardware leading to portability.

- Reduction in power requirements leading to portability.
- New display technologies leading to the packaging of computational devices in new forms.
- Assimilation of computation into the environment (e.g., VCRs, microwave ovens, televisions).
- Specialized hardware leading to new functions (e.g., rapid text search).
- Increased development of network communication and distributed computing.
- Increasingly widespread use of computers, especially by people who are outside of the computing profession.
- Increasing innovation in input techniques (e.g., voice, gesture, pen), combined with decreasing costs, leading to rapid computerization for people previously left out of the "computer revolution".
- Wider social concerns leading to improved access to computers by currently disadvantaged groups (such as young children or the physically/visually disabled).

According to Hewett et al., one consequence of the above developments is that "computing systems will appear partially to dissolve into the environment and become much more intimately associated with their users' activities". Obviously, this development quintessentially is congruent with the anticipated effects and implications of emerging ubiquitous computing systems and environments.

B.4.2. General Research Challenges

Based on the described trends, Hewett et al. expect HCI research challenges in the following fields [HBC+96]:

- *Anytime, anywhere accessibility of services by means of ubiquitous communication.* Computers will communicate through high speed local networks, nationally over wide-area networks, and portably via infrared, ultrasonic, cellular, and other technologies. Data and computational services will be portably accessible from many if not most locations to which a user travels.
- *High functionality systems.* Systems will have large numbers of functions associated with them. There will be so many systems that most users, technical or non-technical, will not have the time to learn them in the traditional way (e.g., by reading thick manuals).
- *Mass availability of computer graphics.* Computer graphics capabilities such as image processing, graphics transformations, rendering, and interactive animation will become widespread as inexpensive chips become available for inclusion in general workstations.

- *Mixed media systems.* Systems will handle images, voice, sounds, video, text, formatted data. These will be exchangeable over communication links among users. The separate worlds of consumer electronics (e.g., stereo sets, VCRs, televisions) and computers will partially merge. Computer and print worlds will continue to cross assimilate each other.

- *Interfaces for high-bandwidth interaction.* The rate at which humans and machines interact will increase substantially due to the changes in speed, computer graphics, new media, and new input/output devices. This will lead to some qualitatively different interfaces, such as virtual reality or computational video.

- *Large and thin displays.* New display technologies will finally mature enabling very large displays and also displays that are thin, light weight, and have low power consumption. This will have large effects on portability and will enable the development of paper-like, pen-based computer interaction systems very different in feel from desktop workstations of the present.

- *Novel means of embedded computation.* Computation will pass beyond desktop computers into every object for which uses can be found. The environment will be alive with little computations from computerized cooking appliances to lighting and plumbing fixtures to window blinds to automobile braking systems to greeting cards. To some extent, this development is already taking place. The difference in the future is the addition of networked communications that will allow many of these embedded computations to coordinate with each other and with the user. Human interfaces to these embedded devices will in many cases be very different from those appropriate to workstations.

- *Group interfaces.* Interfaces to allow groups of people to coordinate will be common (e.g., for meetings, for engineering projects, for authoring joint documents). These will have major impacts on the nature of organizations and on the division of labor. Models of the group design process will be embedded in systems and will cause increased rationalization of design.

- *User tailorability.* Ordinary users will routinely tailor applications to their own use and will use this power to invent new applications based on their understanding of their own domains. Users, with their deeper knowledge of their own knowledge domains, will increasingly be important sources of new applications at the expense of generic systems programmers (with systems expertise but low domain expertise).

- *Information utilities.* Public information utilities (such as home banking and shopping) and specialized industry services (e.g., weather for pilots) will continue to proliferate. The rate of proliferation will accelerate with the introduction of high-bandwidth interaction and the improvement in quality of interfaces.

The appearance of ubiquitous computing technologies provides the technical foundations for advanced input and output devices as well as for ever more sophisticated small-scale embedded systems. Consequently, the design space of human-

computer interfaces, in which physical objects play a central role as physical representations and controls for digital information, grew accordingly. This led to a "wave of new HCI research into ways to link the physical and digital worlds" [UI00], aiming at exploring the relationship between physical representation and digital information, and highlighting new kinds of interaction that are not readily described by existing frameworks. These interfaces are typically referred to as *graspable interfaces* [FIB95] or *tangible user interfaces* [IU97].

The challenge of tangible user interfaces is the seamless integration of representation and control, which differs markedly from the mainstream graphical user interface (GUI) approaches of modern HCI. While graphical interfaces make a fundamental distinction between input devices (such as the keyboard and mouse as controls) and graphical output devices (such as monitors and head-mounted displays) for the synthesis of visual representations, tangible interfaces explore the conceptual space opened by the elimination of this distinction.

B.5. Context-Aware Computing

B.5.1. Definition of Context

When humans talk with humans, they are in a position to use *implicit* situational information, or context, to increase the conversational bandwidth [Dey01]. This is possible for several reasons, such as because of the richness of the language they share, their common understanding of how the world works, and owing to an implicit understanding of everyday situations. Context-aware computing likewise aims at enabling computer systems to gain a certain knowledge and understanding about the particular situation of a person or a device, and to use this knowledge as a source for implicit input to improve the quality and adaptiveness of computing service provided to the user. In short, context-aware computing can be characterized as the ability of a software system to continuously adapt its behavior to a changing environment over which it has little or no control [RJH02].

There have been many attempts to define the terms "context" and "context awareness" more precisely. Dey and Abowd [DA00], for instance, define *context* as "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 applications themselves." Similarly, Dey and Abowd define a system to be *context-aware* "if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task." Typically, relevant context information is limited to the physically proximate environment of a user or device, since for context-aware systems, the "interesting part of the world around us is what we can see, hear, and touch" [SAW94]. Dey et al. also proposed a categorization which combines ideas from previous taxonomies and attempts to generalize them so that they satisfy a broader range of existing context-aware applications. They discern three categories of context: *computing context* (locally available resources, network connectivity, communication bandwidth, etc.), *physical context* (physical properties of the user's environment, such as temperature, humidity, lighting, noise level, etc.), and *user context* (e.g., the user's profile, location, nearby people, the user's activity, etc.). They further describe three types of features that a context-aware application can

support: presentation of information and services to a user, automatic execution of a service for a user, and tagging of context to information to support later retrieval.

Obviously, context awareness is a challenge for mobile devices whose location and environment is liable to change frequently. Here, context-aware computing applications can promote and mediate people's interactions with devices, computers, and other people, and it can help to navigate in unfamiliar places [SAW94]. Schilit et al. have classified context-aware applications into four major categories: proximate selection as an interface technique facilitating the interaction with nearby objects, automatic contextual reconfiguration of system components, contextual information and commands, and context-triggered actions [SAW94].

B.5.2. Situation Awareness

Even the availability of context information does not imply its usefulness, especially if it is provided in an unstructured or unorganized fashion. For context information to become useful, it is important that it can be used to draw conclusions on the actual activity, intention, or situation of a user. Therefore, the need arises for a *situation abstraction* [Dey01] that provides a description of entities relevant to assess the situational context of a single entity.

Situation awareness can be considered the knowledge of all the objects that are relevant to a subject and the knowledge of the relations that the subject has with these objects. According to Kokar [Kok04], a subject is *aware* if the subject not only observes (experiences) the objects but also is capable of drawing conclusions from these observations. Kokar further states that being aware of a specific situation means to be aware of the way in which something is placed in relation to its surroundings. Kokar points out the fact that while situation awareness clearly requires an inference capability, the inference process itself can be carried out using either a mathematical or a logical framework. In doing so, mathematical and logical reasoning are dual techniques: Logical reasoning can be viewed as the application of inference rules on a syntactical level, yielding a good model for the automation of the inference process. Mathematical reasoning relies more on the ingenuity of the human to derive proofs. It can be viewed as a process of manipulating the semantic objects rather than the syntactic objects, relying more explicitly on models (i.e., sets, functions, and relations) than it is the case in logical reasoning. To make use of situation awareness, one must be able to recognize situations, assess their impact on one's goals, memorize situations, associate various properties with specific situations, and communicate descriptions of situations to others.

B.5.3. Location Awareness

Context consisting of any information that (1) can be used to characterize the particular location of an entity, or that (2) is characteristic of a particular location, is commonly referred to as *location context*. Naturally, location context subsumes information on the user's physical position (e.g., global GPS coordinates) as well as information about symbolic locations which capture more abstract ideas of where something is (such as in the kitchen, in the office, on a train, etc.). Such location context can directly be used for the localization and tracking of objects or people, for creating geographic or topological maps, and for the implementation of

navigation systems in general.

Indirectly, location context is used for providing location-based services which have the goal of delivering location-aware content to subscribers on the basis of the positioning capability of the wireless infrastructure [CCR+04]. This includes applications that provide location-contextual information and commands, such as location-aware tour guides [DCME01] which offer information on nearby points of interest, or information systems for visually impaired people [CKR04] that describe the user's immediate location-dependent surroundings, or location-aware services that allow the user to use the nearest printer, indicate the closest restaurant, or guide a person to the nearest emergency exit.

Besides, location context often provides valuable input for determining the situational context of a user, allowing mobile devices to respond to changes in the location context and to adapt their functionality accordingly, thus making them *location-aware*. For instance, by knowing and understanding the user's symbolic location, a smart phone could be configured to ring loudly in outdoor environments but to remain silent in the cinema, provided that the device has the means of determining its location context in some way.

B.5.4. General Research Challenges

Support for context awareness can be provided indirectly to devices by means of suitable background infrastructures. Such infrastructures often possess a global view on the location of users and equipment, and on the capabilities of the equipment and networking infrastructure. In this case we speak of *indirect* context awareness, where the entire sensing and processing occurs in the infrastructure while the mobile device obtains its context by means of communication [GSB02]. For instance, Harter et al. [HHS+02] describe a sensor-driven platform for context-aware computing that collects environmental data in a form suitable for context-aware services, enabling applications to follow mobile users as they move around in a building. Assuming a richly equipped networked environment such as a modern office, the users no longer need to physically carry any equipment with them as the user-interfaces of the applications themselves can follow the users and exploit the locally available equipment and networking resources (*follow-me* applications). Further examples of context-aware applications are self-organizing computing systems (which Harter et al. called "construction-kit computers") that automatically build themselves by organizing nearby components to act as a more complex device, or "walk-through videophones" which automatically select streams from a range of cameras to maintain an image of a nomadic user [HHS+02].

In contrast to indirect context-awareness, a device has *direct* context awareness if it is able to obtain context autonomously and independently from background infrastructures [GSB02]. Portable or wearable computers particularly benefit from direct context or situation-awareness, as they have the potential "to 'see' as the user sees, 'hear' as the user hears, and experience the life of the user in a 'first-person' sense" [SSR+99]. By providing contextual information, a challenge is to develop more "intelligent and fluid interfaces that use the physical world as part of the interface". Wearable computers can perform useful work even while the wearer isn't directly interacting with the system, or in environments where the user needs to concentrate on his environment instead of on the computer interface. For this

the wearable device needs to use information from the wearer's context (intention, location, activity) to be the least distracting. For example, a location-aware notification system can adjust its means of user notification (using an auditory, visual, or haptic signal, or suppressing the notification altogether) depending on the user's location (office, home, subway) or on his activity (meeting, conversation).

To infer the user's situational context, especially the user's current intention or activity, by means of diverse sensors is a difficult problem. Kokar has identified a number of general technical challenges for research on situation awareness that focus on aspects of information fusion and reasoning [Kok04]:

- *Situations as objects:* Situation awareness in Level 2 fusion[1] is considered a computer-based process that can recognize and manipulate situations, and the computer is also assumed to be responsible for making decisions depending on the situation. According to Kokar, this issue is different from human-oriented situation awareness (i.e., the human becomes aware of a situation and then uses this awareness for decision making), and it creates many challenges that need to be addressed by the developers of such computer systems.

- *Relation derivation algorithms:* The question whether a particular relation holds or not is non-trivial for a program to answer. Answers to the question can either be remembered as an atomic fact, or determined by applying a rule to establish such a fact. In general, for each relation a rule is needed to derive its validity, and for any domain a corresponding set of rules has to be developed.

- *Relevance of relations:* If the number of "situation objects" is large, the number of potential valid relations potentially grows exponentially. An important challenge is to define methods that allow the system to determine which of the possible relations are relevant and should be derived or monitored.

- *Complexity of derivation algorithms:* Relations can be derived either using a declarative (or logical) or a procedural (or mathematical) approach. The declarative approach facilitates the specification of all the facts that the derivation algorithm should consider. Conceptually, declarative derivation algorithms are simple search algorithms, but with exponential search complexity. Hence, the challenge is to find algorithms that can assess their own limitations and abilities given the resources. The procedural approach, on the other hand, requires the specification of domain-specific derivation procedures already during the design of a situation awareness system. Here the challenge is for the designer to develop a relatively complete set of such procedures.

- *Certainty of derived relations:* An immense challenge is the certainty of derived decisions, independently of the chosen approach to determine the relations. Especially since in practice, it is highly unlikely that the designer of a situation awareness system will be in a position to provide a complete coverage of all relations that will potentially be needed by the user of such a system. Incorporating means to compute the uncertainty of decisions adds

[1] See revisions to the JDL data fusion model [HL01]: Level 1 fusion takes raw sensory data and generates either object IDs or object states. Level 2 takes object information from Level 1 and derives relations among the objects.

to the complexity of this task, which remains a great challenge despite many documented attempts to tackle this issue, such as by combining Bayesian reasoning with logical reasoning, for instance.

However, apart from information fusion aspects, it is important to fundamentally understand what context is and how it can be used [Dey01] in order to use it effectively. A major challenge is to provide suitable *architectural support* that enables and facilitates the development of context-aware applications and services. An understanding of context is essential for application designers to be able to choose what context to use in their applications, while an understanding of how context can be used is indispensable for being able to decide which context-aware behaviors to support in the applications. For that, Dey suggested architectural support in form of a toolkit [DAS01], which enables designers to build their applications more easily. On the technical level, once context information is made available to mobile entities, the latter have to be able to understand and correctly interpret this context information in order to use it in a meaningful way. Therefore, a further important challenge is the development of suitable *representations and ontologies* for the sharing and semantic integration of context information [Noy04, WVG04].

Besides technical challenges, the realization of context- and location-aware services in general, which potentially enable computer systems to infer information about people's intentions, activities, and situations, raises a number of social, ethical, and political questions with respect to *privacy* issues [Lan01, Lan02]. For instance, this includes questions such as which personal information a system is allowed to derive without a user's consent, who has access to this information, and how a user can restrict and control the aggregation of sensitive personal information to prevent the formation of so-called information asymmetries [BCL+04a].

B.6. Sensor Networks

B.6.1. Definition

The recent advances in wireless communications and electronics have enabled the development of low-cost, low-power, multi-functional *sensor nodes* that are small in size and communicate wirelessly over short distances. These tiny sensor nodes typically consist of sensing, data processing, and communicating components.

By densely deploying a large number of sensor nodes inside or very close to a phenomenon that has to be observed ("sensed") by those sensors, we obtain a so-called *sensor network*. Usually the position of sensor nodes is not predetermined, which allows a random deployment in inaccessible terrains or disaster relief operations. For this reason, sensor network protocols and algorithms must possess certain self-organizing capabilities. Furthermore, a unique feature of sensor networks is the support of cooperation among the single sensor nodes. A typical task of sensor nodes is the fusion of data received from other nodes. Since sensor nodes are fitted with an on-board processor, they are able to locally perform simple computations and transmit only the required and partially processed data to other nodes instead of sending the raw data, which helps to reduce the communication load and energy consumption in the sensor network as a whole.

Sensor network applications usually require *wireless ad hoc networking* techniques for the coordination of cooperative activities. Akyildiz et al. [ASSC02a]

showed that many protocols and algorithms that have been proposed for traditional wireless ad hoc networks are not well suited to the unique features and application requirements of sensor networks. To illustrate this point, they described a number of differences between sensor networks and ad hoc networks:

- The number of sensor nodes in a sensor network can be several orders of magnitude higher than the nodes in an ad hoc network.
- Sensor nodes are densely deployed.
- Sensor nodes are prone to failures.
- The topology of a sensor network changes very frequently.
- Sensor nodes mainly use a broadcast communication paradigm, whereas most ad hoc networks are based on point-to-point communications.
- Sensor nodes are limited in power, computational capacities, and memory.
- Sensor nodes may not have global identification (ID) because of the large amount of overhead and the large number of sensors.

Research in the area of sensor networks is therefore concerned with the development of schemes that fulfill these requirements.

B.6.2. Design Space for Sensor Networks

The design space for sensor networks comprises the following parameters [ASSC02a]:

- *Fault tolerance:* Sensor nodes are liable to fail or to be blocked due to physical damage, environmental interference, or lack of power. Since the failure of single sensor nodes should not affect the overall task of the sensor network as a whole, fault tolerance is required. In this context, fault tolerance can be defined as the ability to sustain sensor network functionalities without any interruption due to sensor node failures [HaAA00, SSJ01].
- *Scalability:* The number of sensor nodes deployed for the monitoring of a phenomenon may be on the order of hundreds, thousands, or even millions, depending on the particular application. Sensor networking schemes must therefore be able to work with such high numbers of nodes and with high node densities.
- *Production costs:* Due to the potentially large number of sensor nodes, the cost of a single node is very important to justify the overall cost of the network. Consequently, a sensor network is only "cost-justified" if the cost of the network is less expensive than deploying traditional sensors. The cost of a sensor node should be substantially less than US-$1 in order for large-scale sensor networks to become economically feasible [RAdS+00].
- *Hardware constraints:* A sensor node is typically made up of four basic components: a sensing unit, a processing unit, a transceiver unit, and a power unit (which may be supported by power scavenging units such as solar cells,

for instance). It may further contain certain application-specific components. Since many network routing techniques and sensing tasks require knowledge of location with high accuracy, it is also common that sensor nodes possess an integrated location finding system. If sensor nodes have to be able to autonomously move to a different location, they need a so-called mobilizer. All these components have to comply with the prevailing hardware constraints, such as stringent size and power consumption constraints, low production cost, the ability to be dispensable, operate without supervision, and to be adaptive to changes in the environment.

- *Sensor network topology:* Since thousands of sensor nodes may have to be deployed throughout a sensor field, the obtained node densities may be as high as 20 nodes per cubic meter [SCI+01]. Such a dense deployment of a high number of nodes requires a deliberate topology maintenance procedure, which is typically divided into three phases: *pre-deployment and deployment phase* (where sensor nodes are either thrown in as a mass or placed one by one in the sensor field), *post-deployment phase* (where topology changes occur due to changes in sensor nodes, position, reachability (due to jamming, noise, moving obstacles, etc.), available energy, malfunctioning, and task details), and *redeployment of additional nodes phase* (additional sensor nodes can be redeployed at any time to replace malfunctioning nodes or due to changes in task dynamics).

- *Environment:* As a rule, it is necessary to densely deploy sensor nodes either very close to or directly inside the phenomenon that has to be observed. Therefore, the distributed sensor nodes usually have to work unattended in remote geographic areas, such as at the bottom of an ocean, in a biologically or chemically contaminated field, in a battlefield beyond the enemy lines, or in a home or large building, for instance. Each of these environments is liable to have its own specific constraints and boundary conditions that the sensor nodes have to be in a position to cope with (e.g., high levels of temperature, pressure, humidity, and signals or materials interfering with the wireless transmissions of the sensor nodes).

- *Transmission media:* In multi-hop sensor networks, communicating nodes are linked by means of a wireless medium. These links can be formed by radio, infrared, or optical media, for instance. To enable global operation of these networks, the chosen transmission medium has to be available worldwide.

- *Power consumption:* Wireless sensor nodes are small-sized microelectronic devices that can only be equipped with a limited power source. As a rule, the lifetime of a sensor node therefore strongly depends on battery lifetime, especially in application scenarios where the replenishment of power resources may not be an option. Further, in multi-hop ad hoc sensor networks, the malfunctioning of a few nodes can cause significant topological changes that may require rerouting of packets and reorganization of the network, leading to an increased energy consumption. Therefore, the conservation of energy and the application of power management mechanisms during sensing, communication, and data processing is a major requirement. A related challenge is the design of power-aware protocols and algorithms for sensor networks

that minimize the overall power consumption. Such protocols may result in a faster depletion of the energy resources of single sensor nodes compared to other nodes, in order to ensure that the sensor network as a whole remains operational as long as possible.

Sensor networks were originally motivated by military applications, such as for the acoustic surveillance of oceans, for ground target detection [CK03], or more recently for providing infrastructure security as a protection against terrorist threats. However, the low production costs now enable many applications in civilian areas, such as for environment and habitat monitoring, industrial sensing, or traffic control, for example.

B.6.3. General Research Challenges

A number of hard research problems and technical challenges in the field of sensor networks have been identified by Chong and Kumar [CK03]. Apart from the considerable technical problems that sensors networks in general have to deal with in the areas of data processing, communication, and sensor management, they recognized a number of hard problems and technical challenges that stem from the potentially harsh, uncertain, and dynamic environments in which sensor nodes are often deployed, and from the particular energy and bandwidth constraints that have to be met. In addition to the challenges posed by wireless ad hoc networks, they describe additional technical challenges in the following areas:

- *Ad hoc network discovery:* For sensor nodes to operate properly, a certain level of knowledge of the sensor network is essential. For instance, sensor nodes need to know their own location and the identity and location of their neighbors to support processing and collaboration.

- *Network control and routing:* A sensor network has to be able to operate autonomously, which requires that it is in a position to dynamically adjust its behavior and configuration to dynamically changing resources (e.g., energy, bandwidth, processing power). Since connectivity cannot be taken for granted in ad hoc networks, it has to "emerge" as needed from the algorithms and software. Further, as communication links are unreliable, the software and system design should generate the required reliability, which requires research into issues such as network size and the number of links and nodes needed to provide adequate redundancy.

  Survivability and adaptability to the environment constitute particular challenges of sensor networks with regard to communication and routing. These challenges need to be addressed by deploying an adequate number of nodes to provide redundancy in paths, and by providing algorithms that find the right paths. A further possible solution is the development of diffusion routing methods, which rely only upon information found at neighboring nodes [EGHK99]. However, such methods may not be optimal in so far that they don't achieve the information-theoretic capacity of a spatially distributed wireless network [GK00]. Another important design challenge is the investigation of how system parameters, such as network size and density of nodes per square mile, affect the trade-offs between latency, reliability, and energy.

- *Collaborative signal and information processing:* The main purpose of sensor nodes in an ad hoc sensor network is to collaborate in order to collect and process data to generate useful information. In this context, the collaborative signal and information processing over a network is a new and challenging area of research with links to distributed information fusion. According to Chong and Kumar, important technical issues include the degree of information sharing between nodes, and algorithms for fusing the information from other nodes. Processing data from a higher number of sensors generally results in better performance but also requires more communication resources and causes a higher energy consumption. Similarly, less information is lost when information is communicated at a lower level (e.g., raw signals and data), but at the cost of higher bandwidth requirements. Therefore, a further research challenge is the consideration of the multiple trade-offs between performance and resource utilization in collaborative signal and information processing using microsensors. Further challenges are correct data association, which is an important problem when multiple targets are present in a small region, the enforcement of latency and reliability requirements, and the maximization of the operational life of a sensor network.

- *Tasking and querying:* Chong and Kumar consider a sensor field to be "like a database with many unique features", with the difference that data is dynamically acquired from the environment instead of being entered by an operator. Here an important challenge is to provide a simple interface to users that enables them to interactively task and query the sensor network, even when the data is distributed across a large number of nodes that are geographically dispersed and connected by unreliable links. Such a database view is the more challenging for military applications, where low-latency, real-time, and high-reliability constitute stringent requirements of the battlefield. Further challenges are the development of efficient distributed mechanisms for query and task compilation and placement, data organization, and caching. Also, since mobile platforms can carry sensors and query devices, seamless internetworking between mobile and fixed devices in the absence of any infrastructure support is a critical and unique requirement for sensor networks. For example, this would allow a mobile device to initiate a query in one physical location and then tell the sensor network that it expects to exfiltrate the response to the query when it passes a different specific location after some time.

- *Security:* Sensor nodes may operate in a hostile environment and are liable to attempts of spying and tampering. As a consequence, if the integrity and confidentiality of data generated by and transmitted among sensor nodes is essential, security measures have to be taken into account already at the design stage. Research challenges include the investigation of network techniques for low-latency, survivable, and secure networks, and the protection against illegitimate intrusions and spoofing. In military scenarios where sensors are being envisioned for use behind enemy lines, a critical challenge is to keep the probability of detecting communication among sensor nodes as low as possible.

B.6.4. Dependability as a System-Centric Challenge

Sensor networks in particular owe their success to the advances in microelectronics and embedded computing, as it is the case with ubiquitous computing systems in general. Further, from a technological point of view, a sensor node is not fundamentally different from a smart object found in a ubiquitous computing environment, as described in Section B.3.3.

However, the research discipline of sensor networks takes on a special position in ubiquitous computing: it promotes a *system-centric* view that is fundamentally different from the *user-centric* view that prevails in ubiquitous computing scenarios in general. On the one hand, this fact is mirrored in the nature of sensor network research challenges, which are strongly influenced by technical, system-oriented aspects, such as concerning transmission media, communication protocols, routing algorithms, low-level data fusion, collaborative signal processing, topology management issues [CK03]. The system-centricity is also evident in explicitly system-centric research topics, such as the study of the information-theoretic capacity of a spatially distributed wireless network [GK00], for example.

Typical system-centric goals of sensor network research are the optimization of the lifetime and/or performance of a deployed sensor network as a whole. For example, individual sensor nodes may be sacrificed in order to minimize the overall net energy consumption, or to maximize areal coverage and inter-node-connectivity in the monitored area. The interests of an individual sensor node take second place to system-wide objectives. Here the needs of individual users play only an insignificant role: usually there is no one-to-one relationship or explicit interaction between users and individual sensor nodes, whereas this is often the case with smart objects and handheld devices in ubiquitous computing environments.

Likewise, dependability concepts for sensor networks concentrate on technical, system-specific issues, such as robust and fault-tolerant communication [IGE00, ABL04], routing [HMKR04, TZVM04], and collaboration [PK00]. In doing so, fundamental sensor network aspects such as scalability [BEGH01, WOW+05], energy efficiency [PK00, EKM04, KPS04, BFA05], coordination [EGHK99], self-organization capabilities [BEGH01, WOW+05] and data retrieval techniques [GGE+05] are investigated from a system-centric viewpoint.

The dependability of the embedded sensor nodes themselves is largely determined by the correct specification and implementation of their operating software, and by their particular hardware, which often exhibits the tendency to degrade and fail over time [Kni02]. Here traditional dependability concepts such as replication of processes [JKH05] or hardware components [Rom04], and techniques for safe software development [Lev02] are applicable.

The dependability challenges of sensor networks are beyond the scope of this dissertation, as we focus on novel *user-centric* dependability challenges that arise with the appearance of ubiquitous computing systems and environments.

C. About the Author

Jürgen Josef Bohn received a Master's Degree (Diplom) in Computer Science from the University of Karlsruhe (TH), Germany, in 2000. He completed his Master's Thesis in the field of Mobile Agent Security at the IBM Zurich Research Laboratory in Rüschlikon, Switzerland. From 2000 until 2006 he was a Research Assistant in the Distributed Systems Group at the Institute for Pervasive Computing at ETH Zurich, Switzerland, where he obtained his Doctoral Degree in Computer Science. Since August 2006 he has been a Technical Project Coordinator and Researcher at the Wernher von Braun Center for Advanced Research in Campinas, Brazil, where he is currently working on innovative Radio Frequency Identification (RFID) systems and applications.

Contact: jjbohn@jjbohn.com

Bibliography

[AAA97] Ward Andy, Jones Alan, and Hopper Andy. A New Location Technique for the Active Office. *IEEE Personal Communications*, 4(5):42–47, 1997.

[AAC+00] Harold Abelson, Don Allen, Daniel Coore, Chris Hanson, George Homsy, Thomas F. Knight, Jr., Radhika Nagpal, Erik Rauch, Gerald Jay Sussman, and Ron Weiss. Amorphous Computing. *Communications of the ACM*, 43(5):74–82, March 2000.

[ABL04] Giuseppe Anastasi, Alberto Bartoli, and Flaminia L. Luccio. Fault-tolerant support for reliable multicast in mobile wireless systems: design and evaluation. *Wireless Networks*, 10(3):259–269, 2004.

[ABO02] Gregory D. Abowd, Agathe Battestini, and Thomas O'Connell. The Location Service: A framework for handling multiple location sensing technologies, 2002. Available online at `http://www.cc.gatech.edu/fce/ahri/publications/location_service.pdf`.

[ACDP05] Matthieu Anne, James L. Crowley, Vincent Devin, and Gilles Privat. Localisation intra-batiment multi-technologies: RFID, wifi et vision. In *UbiMob '05: Proceedings of the 2nd French-speaking conference on Mobility and ubiquity computing*, pages 29–35, New York, NY, USA, 2005. ACM Press.

[ACH+01] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper. Implementing a Sentient Computing System. *Computer*, 34(8):50–56, 2001.

[ACT00] Marcos Kawazoe Aguilera, Wei Chen, and Sam Toueg. Failure Detection and Consensus in the Crash Recovery Model. *Distributed Computing*, 13(2):99–125, April 2000.

[Ada00] Natascha Adamowsky. Kulturelle Relevanz. Ladenburger Diskurs "Ubiquitous Computing", February 2000. Available online at `http://www.inf.ethz.ch/vs/events/slides/adamowldbg.pdf`.

[ADS02] James Aspnes, Zoë Diamadi, and Gauri Shah. Fault-tolerant routing in peer-to-peer systems. In *Twenty-First ACM Symposium on Principles of Distributed Computing*, pages 223–232, July 2002.

[AJLS97] M. D. Addlesee, A. Jones, F. Livesey, and F. Samaria. The ORL Active Floor. *IEEE Personal Communications*, 4(5):35–41, October 1997.

[AKS04] Ankur Agiwal, Parakram Khandpur, and Huzur Saran. LOCATOR: location estimation system For wireless LANs. In *WMASH '04: Proceedings of the 2nd ACM international workshop on Wireless mobile applications*

and services on WLAN hotspots, pages 102–109, New York, NY, USA, 2004. ACM Press.

[AL86] A. Avizienis and J.-C. Laprie. Dependable computing: From concepts to design diversity. *Proceedings of the IEEE*, 74(5):629–638, May 1986.

[AM00] Gregory D. Abowd and Elizabeth D. Mynatt. 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.

[Ana05] Analysys Research. The Western European Mobile Market: trends and forecasts 2005-2010. 6th Edition, April 2005. Available online at `http://research.analysys.com/default.asp?Mode=article\&iLeftArticle=1876`.

[Ano01] Anonymous. Workshop on Embedded Computing. Report, Sponsored by NSF and ACM SIGDA, November 2001.

[Ara95] Agustin A. Araya. Questioning Ubiquitous Computing. In *Proceedings of the 1995 ACM 23rd Annual Conference on Computer Science*. ACM Press, 1995. Available online at `http://doi.acm.org/10.1145/259526.259560`.

[ART05] O. Amft, J. Randall, and G. Tröster. Towards LuxTrace: Using solar cells to support human position tracking. In *Proceedings of 2nd International Forum on Applied Wearable Computing*, Zürich, Switzerland, March 2005.

[AS85] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21:181–185, 1985.

[ASRPZ03] Stefan Arbanowski, Stephan Steglich, Ilja Radusch, and Radu Popescu-Zeletin. Super Distributed Objects: An Execution Environment for I-Centric Services. In *Proceedings 9th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS' 03)*, pages 201–208. IEEE Computer Society, October 2003.

[ASSC02a] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002.

[ASSC02b] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *IEEE Computer Networks*, 38(4):393–422, March 2002.

[ATS04] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371, 2004.

[Bak04] Bonnie C. Baker. Wireless Communication Using the IrDA Standard Protocol. Analog Design Note ADN006, Microchip Technology Inc, January 2004.

[Bal96] T. R. Balch. Grid-based navigation for mobile robots. *The Robotics Practitioner*, 2(1), 1996.

[Bar02] David Barkai. *Peer-to-Peer Computing. Technologies for Sharing and Collaborating on the Net*. Rich Bowles, 2002.

[Bär04] Marco Bär. Collaborative Map-Making in an Area of Randomly Distributed RFID-Tags using a LEGO Mindstorms Robot Vehicle. Semester thesis, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, July 2004.

[BBH+04] Gaetano Borriello, Waylon Brunette, Matthew Hall, Carl Hartung, and Cameron Tangney. Reminding About Tagged Objects Using Passive RFIDs. In *Proceedings of 6th International Conference on Ubiquitous Computing (UbiComp 2004), Nottingham, UK, September 7-10, 2004*, number 3205 in Lecture Notes in Computer Science, pages 36–53. Springer, 2004.

[BC06] Bluetooth Consortium. Bluetooth.com – The Official Bluetooth Wireless Info Site. `www.bluetooth.com`, February 2006.

[BCL+03a] Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, and Michael Rohs. Allgegenwart und Verschwinden des Computers – Leben in einer Welt smarter Alltagsdinge. In Ralf Grötker, editor, *Privat! Kontrollierte Freiheit in einer vernetzten Welt*, pages 195–245. Heise-Verlag, March 2003.

[BCL+03b] Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, and Michael Rohs. Disappearing Computers Everywhere – Living in a World of Smart Everyday Objects. In *New Media, Technology and Everyday Life in Europe Conference (EMTEL 2003)*, London, UK, April 2003. Available online at `http://www.emtelconference.org/`.

[BCL+04a] Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, and Michael Rohs. Living in a World of Smart Everyday Objects – Social, Economic, and Ethical Implications. *Journal of Human and Ecological Risk Assessment*, 10(5), October 2004.

[BCL+04b] Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, and Michael Rohs. Social, Economic, and Ethical Implications of Ambient Intelligence and Ubiquitous Computing. In W. Weber, J. Rabaey, and E. Aarts, editors, *Ambient Intelligence*. Springer, 2004.

[BDM93] Michael Barborak, Anton Dahbura, and Miroslaw Malek. The consensus problem in fault-tolerant computing. *ACM Computing Surveys*, 25(2):171–220, June 1993.

[BEGH01] Nirupama Bulusu, Deborah Estrin, Lewis Girod, and John Heidemann. Scalable coordination for wireless sensor networks: Self-configuring localization systems. In *Proceedings 6th International Symposium on Communication Theory and Applications (ISCTA '01)*. University of California, Los Angeles, July 15-20th 2001.

[BFA05] Azzedine Boukerche, Xin Fei, and Regina B. Araujo. An energy aware coverage-preserving scheme for wireless sensor networks. In *PE-WASUN '05: Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 205–213, New York, NY, USA, 2005. ACM Press.

[BFH97] W. Burgard, D. Fox, and D. Hennig. Fast Grid-based Position Tracking for Mobile Robots. In *Proceedings of the 21th German Conference on Artificial Intelligence*, volume 1303 of *Lecture Notes in Computer Science*. Springer, 1997.

[BG95] Kenneth P. Birman and Bradford B. Glade. Reliability through consistency. *IEEE Software*, pages 29–41, May 1995.

[BGS01] M. Beigl, H.-W. Gellersen, and A. Schmidt. Mediacups: Experience with Design and Use of Computer-Augmented Everyday Artefacts. *Computer Networks, Special Issue on Pervasive Computing, Elsevier*, 35(4):401–409, March 2001.

[BGV02] Jürgen Bohn, Felix Gärtner, and Harald Vogt. Dependability Issues of Pervasive Computing in a Healthcare Environment. In Dieter Hutter, Günter Müller, Werner Stephan, and Markus Ullmann, editors, *Proceedings of the 1st International Conference on Security in Pervasive Computing*, number 2082 in Lecture Notes in Computer Science, Boppard, Germany, March 2002. Springer.

[BH00] Levente Buttyan and Jean-Pierre Hubaux. Enforcing Service Availability in Mobile Ad-Hoc WANs. In *IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2000.

[BH01] Levente Buttyan and Jean-Pierre Hubaux. Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks. Technical report, EPFL Lausanne, Switzerland, 2001.

[BH03] Levente Buttyán and Jean-Pierre Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *Mobile Networks and Applications*, 8(5):579–592, 2003.

[Bha97] Vaduvur Bharghavan. Challenges and solutions to adaptive computing and seamless mobility over heterogeneous wireless networks. *Wireless Personal Communications*, 4(2):217–256, 1997.

[BHE00] Nirupama Bulusu, John Heidemann, and Deborah Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, October 2000.

[Bir93] Kenneth P. Birman. The process group approach to reliable distributed computing. *Communications of the ACM*, 36(12):37–53, 1993.

[Bit05] Bitlaw.com. Domain names disputes, 2005. Available online at `http://www.bitlaw.com/internet/domain.html#disputes`.

[BJ87] K. P. Birman and T. A. Joseph. Reliable Communication in the Presence of Failures. *ACM Transactions on Computer Systems*, 5(1):47–76, February 1987.

[BJM+91] M. Banâtre, Ph. Joubert, Ch. Morin, G. Muller, B. Rochat, and P. Sanchez. Stable transactional memories and fault tolerant architectures. *SIGOPS Operating Systems Review*, 25(1):68–72, 1991.

[BK02] H.-B. Bludau and A. Koop, editors. *Proceedings 2nd Conf. on Mobile Computing in Medicine, Workshop of the Project Group MoCoMed, GMDS-Fachb. Med. Informatik & GI-Fachausschuss 4.7, Heidelberg*, volume 15 of *LNI*. GI, 2002.

[BKM+04] Jan Beutel, Oliver Kasten, Friedemann Mattern, Kay Römer, Frank Siegemund, and Lothar Thiele. Prototyping Wireless Sensor Network Applications with BTnodes. In *1st European Workshop on Wireless Sensor Networks (EWSN)*, pages 323–338, Berlin, Germany, January 2004. Springer.

[BKW02] J. Baus, A. Krüger, and W. Wahlster. A Resource-Adaptive Mobile Navigation System. In *Proceedings of International Conference on Intelligent User Interfaces IUI 2002*. ACM Press, 2002.

[BM04] Jürgen Bohn and Friedemann Mattern. Super-Distributed RFID Tag Infrastructures. In Panos Markopoulos, Berry Eggen, Emile Aarts, and James Crowley, editors, *Proceedings of the 2nd European Symposium on Ambient Intelligence (EUSAI 2004), Eindhoven, The Netherlands*, number 3295 in Lecture Notes in Computer Science, pages 1–12. Springer, November 2004.

[BMF+00] Wolfram Burgard, Mark Moors, Dieter Fox, Reid Simmons, and Sebastian Thrun. Collaborative Multi-Robot Exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[Boh03] Jürgen Bohn. Exploiting Heterogeneity in Ubiquitous Computing Environments for Robust Positioning and Localization. Workshop on Location-Aware Computing at Ubicomp 2003, October 2003.

[Boh04a] Jürgen Bohn. Instant Personalization and Temporary Ownership of Handheld Devices. In *Proceedings of the 6th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2004)*, pages 134–143, Windermere, Cumbria, UK, December 2004. IEEE Computer Society Press, Los Alamitos, California.

[Boh04b] Jürgen Bohn. The Smart Jigsaw Puzzle Assistant: Using RFID Technology for Building Augmented Real-World Games. Workshop on Gaming Applications in Pervasive Computing Environments at PERVASIVE 2004, Vienna, Austria, April 2004. Available online at `http://www.ipsi.fraunhofer.de/ambiente/pervasivegaming/`.

[Boh06] Jürgen Bohn. Prototypical Implementation of Location-Aware Services based on Super-Distributed RFID Tags. In *Proceedings 19th International Conference on Architecture of Computing Systems (ARCS'06)*, number 3894 in Lecture Notes in Computer Science, pages 69–83, Frankfurt am Main, Germany, March 2006. Springer.

[Boh07a] Jürgen Bohn. iPOS: A Fault-Tolerant and Adaptive Multi-Sensor Positioning Architecture with QoS Guarantees. *Sensor Review Journal*, 2007. Accepted for Publication.

[Boh07b] Jürgen Bohn. Prototypical Implementation of Location-Aware Services based on a Middleware Architecture for Super-Distributed RFID Tag Infrastructures. *Personal and Ubiquitous Computing*, 2007. In Press. Online version available at `http://dx.doi.org/10.1007/s00779-006-0107-2`.

[BP00] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM 2000*, pages 775–784, 2000.

[BR01] Jürgen Bohn and Michael Rohs. Klicken in der realen Welt. In *Konferenz Mensch und Computer 2001. Workshop Mensch-Computer-Interaktion in allgegenwärtigen Informationssystemen*. Michael Beigl, Hans-Werner Gellersen, and Norbert Streitz, Bad Honnef, Bonn, March 2001. Available online at `http://www.teco.edu/mc2001/`.

[Bre03] Elizabeth A. Bretz. Precision navigation in European skies. *IEEE Spectrum*, 40(9):16, September 2003.

[BS06] Bluetooth SIG. Bluetooth.org – The Official Bluetooth Membership Site. `www.bluetooth.org`, February 2006.

[Bur05] Martin Burri. LuxTraceRT: A Self-Calibrating Real-Time Positioning System using Solar Cells as Main Sensory Input. Semester thesis, Distributed Systems Group, ETH Zürich, Switzerland, October 2005.

[BV03] Jürgen Bohn and Harald Vogt. Robust Probabilistic Positioning based on High-Level Sensor-Fusion and Map Knowledge. Technical Report 421, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, April 2003.

[BWLW04] Bharat Bhargava, Xiaoxin Wu, Yi Lu, and Weichao Wang. Integrating heterogeneous wireless technologies: a cellular aided mobile ad hoc network (cama). *Mobile Networks and Applications*, 9(4):393–408, 2004.

[Cas02] Henri Casanova. Distributed computing research issues in grid computing. *SIGACT News*, 33(3):50–70, 2002.

[CBM04] Vlad Coroama, Jürgen Bohn, and Friedemann Mattern. Living in a Smart Environment – Scenarios and Implications of the Coming Ubiquitous Information Societyi. In *Proceedings of the International Conference on Systems, Man and Cybernetics 2004 (IEEE SMC 2004)*, volume 6, pages 5633–5638, The Hague, The Netherlands, October 2004.

[CCR+04] Y. Chen, X. Y. Chen, F. Y. Rao, X. L. Yu, Y. Li, and D. Liu. LORE: An infrastructure to support location-aware services. *IBM Journal of Research and Development*, 48(5/6):601–616, 2004.

[CCS99] Dryer D. C., Eisbach C., and Ark W. S. At what cost pervasive? a social computing view of mobile computing systems. *IBM Systems Journal*, 38(4):652–676, 1999.

[CDG+02] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. *SIGOPS Operating Systems Review*, 36(SI):299–314, 2002.

[CDK00] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems - Concepts and Design*. Addison-Wesley, 3rd edition, 2000.

[CGN+05] Jason Campbell, Phillip B. Gibbons, Suman Nath, Padmanabhan Pillai, Srinivasan Seshan, and Rahul Sukthankar. IrisNet: an internet-scale architecture for multimedia sensors. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 81–88, New York, NY, USA, 2005. ACM Press.

[CH04] Vlad Coroama and Norbert Höckl. Pervasive Insurance Markets and their Consequences. In *1st International Workshop on Sustainable Pervasive Computing at PERVASIVE 2004*, Vienna, Austria, April 2004.

[CJ02] P. Chandrasekaran and A. Joshi. MobileIQ: a framework for mobile information access. In *Proceedings 3rd International Conference on Mobile Data Management*, pages 43–50, January 2002.

[CK00] G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Department of Computer Science, Dartmouth College, November 2000.

[CK03] C.-Y. Chong and S. P. Kumar. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, August 2003.

[CKR04] Vlad Coroama, Tarik Kapic, and Felix Röthenbacher. Improving the reality perception of visually impaired through pervasive computing. In Alois Ferscha, Horst Hoertner, and Gabriele Kotsis, editors, *Advances in Pervasive Computing*, pages 369–376, Vienna, Austria, April 2004. Austrian Computer Society (OCG).

[CLMZ03] G. Cabri, L. Leonardi, M. Mamei, and F. Zambonelli. Location-dependent services for mobile users. *IEEE Transactions on Systems, Man and Cybernetics*, 33(6):667–681, November 2003.

[CN02] M. D. Corner and B. D. Noble. Zero-interaction authentication. In *Proceedings 8th Annual International Conference on Mobile computing and networking*, pages 1–11. ACM Press, 2002.

[Cor06] Vlad Coroama. The Smart Tachograph – Individual Accounting of Traffic Costs and its Implications. In *Proceedings of PERVASIVE 2006*, pages 135–152, Dublin, Ireland, May 2006.

[CR03] Vlad Coroama and Felix Röthenbacher. The Chatty Environment – Providing Everyday Independence to the Visually Impaired. Workshop on Ubiquitous Computing for Pervasive Healthcare Applications at UbiComp 2003, October 2003.

[CR04] Vlad Coroama and Felix Röthenbacher. The Chatty Environment. In *Demo at the Second Conference on Pervasive Computing and Communications (PerCom 2004)*, Orlando, Florida, March 2004.

[CRC05] Shiva Chetan, Anand Ranganathan, and Roy Campbell. Towards Fault Tolerant Pervasive Computing. *IEEE Technology and Society*, 24(1):38–44, 2005.

[CSG99] D. Chen, A. Schmidt, and H.-W. Gellesen. An Architecture for Multi-Sensor Fusion in Mobile Environments. In *Proceedings of the International Conference on Information Fusion, Sunnyvale, CA, USA*, volume II, pages 861–868, July 1999.

[DA00] Anind K. Dey and Gregory D. Abowd. Towards a Better Understanding of Context and Context-Awareness. In *Workshop on The What, Who, Where, When, and How of Context-Awareness, Part of the Conference on Human Factors in Computing Systems (CHI 2000), The Hague, The Netherlands*, April 2000.

[DAS01] A. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2–4):97–166, 2001.

[DCME01] N. Davies, K. Cheverst, K. Mitchell, and A. Efrat. Using and Determining Location in a Context-Sensitive Tour Guide. *IEEE Computer*, 34(8):35–41, August 2001.

[Der02] Michael L. Dertouzos. Human-centered systems. In *The invisible future: the seamless integration of technology into everyday life*, pages 181–191. McGraw-Hill, Inc., New York, NY, USA, 2002.

[Dev04] Device Independence Working Group (DIWG). Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation, January 2004.

[Dey01] Anind K. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[DH98] F. Dawson and T. Howes. vCard MIME Directory Profile. Internet RFC 2426, September 1998.

[dI01] Diego López de Ipiña. Video-Based Sensing for Wide Deployment of Sentient Spaces. In *Proceedings of the 2nd PACT 2001 Workshop on Ubiquitous Computing and Communications*, September 2001.

[Dic56] Philip K. Dick. Minority Report. *Fantastic Universe*, January 1956.

[dLKWZ03] E. de Lara, R. Kumar, D. S. Wallach, and W. Zwaenepoel. Collaboration and multimedia authoring on mobile devices. In *Proceedings of MobiSys 2003*, San Francisco, USA, May 2003.

[DM04] F. DePaoli and L. Mariani. Dependability in peer-to-peer systems. *IEEE Internet Computing*, 8(4):54–61, 2004.

[DNH04] Domain Name Handbook. Domain Dispute Index, 2004. Available online at `http://www.domainhandbook.com/dd.html`.

[Dom04] Svetlana Domnitcheva. Smart Vacuum Cleaner – An Autonomous Location-Aware Cleaning Device. In *Adjunct Proceedings of UbiComp 2004*, September 2004.

[DS98] F. Dawson and D. Stenerson. Internet Calendaring and Scheduling Core Object Specification (iCalendar). Internet RFC 2445, November 1998.

[EC03] European Commission. The Galilei Project: GALILEO Design Consolidation. `http://europa.eu.int/comm/dgs/energy_transport/galileo/doc/`, August 2003.

[EC05] European Commission. Galileo project webpage. `http://www.europa.eu.int/comm/dgs/energy_transport/galileo/index_en.htm`, July 2005.

[ECPS02] Deborah Estrin, David Culler, Kris Pister, and Gaurav Sukhatme. Connecting the Physical World with Pervasive Networks. *IEEE Pervasive Computing – Mobile and Ubiquitous Systems*, 1(1):59–69, January 2002.

[EGHK99] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: scalable coordination in sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270. ACM Press, 1999.

[EKM04] Virantha Ekanayake, Kelly Clinton IV, and Rajit Manohar. An ultra low-power processor for sensor networks. In *ASPLOS-XI: Proceedings of the 11th international conference on Architectural support for programming languages and operating systems*, pages 27–36, New York, NY, USA, 2004. ACM Press.

[Ele06] Electronic Frontier Foundation. Blue Ribbon Campaign. Defending Freedom in the Digital World. `www.eff.org/br/`, February 2006.

[Elf89] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer*, 22(6):46–57, June 1989.

[Eur06] Eurocontrol. World Geodetic System 1984. Homepage at `www.wgs84.com`, 2006.

[Fal02] M. A. C. Fallon. Handheld Devices: Toward a More Mobile Campus. *Syllabus Magazine*, November 2002.

[FBDT99] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proceedings of the National Conferenceon Artificial Intelligence*. AAAI Press / The MIT Press, 1999.

[FBKT00] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A Probabilistic Approach to Collaborative Multi-Robot Localization. *Autonomous Robots*, 8(3):325–344, 2000.

[FBT99] D. Fox, W. Burgard, and S. Thrun. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11:391–427, November 1999.

[FHL+03] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian filtering for location estimation. *IEEE Pervasive Computing*, 2(3):24–33, July-September 2003.

[FIB95] George W. Fitzmaurice, Hiroshi Ishii, and William A. S. Buxton. Bricks: laying the foundations for graspable user interfaces. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 442–449. ACM Press/Addison-Wesley Publishing Co., 1995.

[Fin03] K. Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons, 2nd edition edition, April 2003.

[Fit93] George W. Fitzmaurice. Situated Information Spaces and Spatially Aware Palmtop Computers. *Communications of the ACM*, 36(7):38–49, July 1993.

[FJB+04] Richard Fairhurst, Ben Jameson, Andrew Bolt, Robert Brown, and Matthew Slattery. Geowiki - a map which you can annotate. Homepage at `www.geowiki.com`, 2004.

[FKSK02] M. Funabashi, K. Kawano, S. Sameshima, and H. Kato. Middleware technology for ubiquitous computing: Aya (context-aware and yet another service) that permits autonomous collaboration on super distributed objects. In *Proceedings IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 623–628, October 2002.

[FKT01] Ian Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Intl. Journal Supercomputer Applications (IJSA)*, 15(3), 2001.

[FL04] Christian Flörkemeier and Matthias Lampe. Issues with RFID usage in ubiquitous computing applications. In *Proceedings PERVASIVE 2004*, number 3001 in Lecture Notes in Computer Science, pages 188–193, Linz/Vienna, Austria, April 2004. Springer.

[FL05a] Marco Feriencik and Matthias Leumann. iPOS: Ein adaptives System zur Selbst-Positionierung von mobilen ressourcenbeschränkten Geräten. Semester thesis, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, June 2005.

[FL05b] Christian Flörkemeier and Matthias Lampe. RFID middleware design - addressing application requirements and RFID. In *Proceedings sOc-EUSAI 2005 (Smart Objects Conference)*, Grenoble, France, October 2005.

[FLP85] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[FLSC04] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for peer-to-peer networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111, New York, NY, USA, 2004. ACM Press.

[Fos02] Ian Foster. The Grid: A New Infrastructure for 21st Century Science. *Physics Today*, 55(2):42, February 2002.

[FS99] Jason Flinn and Mahadev Satyanarayanan. Energy-aware adaptation for mobile applications. In *SOSP '99: Proceedings of the seventeenth ACM symposium on Operating systems principles*, pages 48–63, New York, NY, USA, 1999. ACM Press.

[FS04] Jason Flinn and Mahadev Satyanarayanan. Managing battery lifetime with energy-aware adaptation. *ACM Transactions on Computer Systems*, 22(2):137–179, 2004.

[FZ94] G. H. Forman and J. Zahorjan. The challenges of mobile computing. *IEEE Computer*, 27(4):38–47, April 1994.

[Gal04] René Gallati. Anpassung eines probabilistischen Positionierungsdienstes für die Verwendung mit mobilen ressourcenbeschränkten Geräten. Semester thesis, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, November 2004.

[Gär99] F. C. Gärtner. Fundamentals of fault-tolerant distributed computing in asynchronous environments. *ACM Computing Surveys (CSUR)*, 31(1):1–26, 1999.

[Gar05a] Gartner, Inc. Gartner Says Strong Fourth Quarter Sales Led Worldwide Mobile Phone Sales to 30 Percent Growth in 2004. Gartner Press Release, 2. March 2005. Available online at `http://www.gartner.com/press_releases/asset_121402_11.html`.

[Gar05b] Gartner, Inc. Gartner Says Wireless E-Mail Applications Drive Worldwide PDA Shipments Increase 25 Percent in First Quarter of 2005. Gartner Press Release, 4. May 2005. Available online at `http://www.gartner.com/press_releases/asset_126216_11.html`.

[GBPK02] J. Glas, M. Banu, V. Prodanov, and P. Kiss. Wireless LANs. In *Proceedings of the Workshop on Advances in Analog Circuit Design*, March 2002.

[GDL+04] Robert Grimm, Janet Davis, Eric Lemar, Adam Macbeth, Steven Swanson, Thomas Anderson, Brian Bershad, Gaetano Borriello, Steven Gribble, and David Wetherall. System support for pervasive applications. *ACM Transactions on Computer Systems*, 22(4):421–486, 2004.

[Geg04] Christian Gegenschatz. Zuverlässige Patientenüberwachung in einer Ubiquitous-Computing-Umgebung. Diplomarbeit, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, June 2004.

[Gel98] Hans-Werner Gellersen. Environment-Mediated Communication. International Workshop on Interactive Applications of Mobile Computing (IMC'98), 1998.

[Ger98] Lou Gerstner. Keynote Speech at CeBIT '98 in Hanover, Germany, 18. March 1998.

[Get93] I. Getting. The Global Positioning System. *IEEE Spectrum 30*, 12:36–47, December 1993.

[GGE+05] Deepak Ganesan, Ben Greenstein, Deborah Estrin, John Heidemann, and Ramesh Govindan. Multiresolution storage and search in sensor networks. *ACM Transactions on Storage*, 1(3):277–315, 2005.

[GHM99] K. Goto, Matsubara H., and S. Myojo. A mobile guide system for visually disabled persons. In *Proceedings 4th International Symposium on Autonomous Decentralized Systems*, pages 12–17, March 1999.

[GK00] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[GKK+03] Phillip B. Gibbons, Brad Karp, Yan Ke, Suman Nath, and Srinivasan Seshan. IrisNet: An Architecture for a World-Wide Sensor Web. *IEEE Pervasive Computing*, October – November:22–33, 2003.

[GLHB03] David Graumann, Walter Lara, Jeffrey Hightower, and Gaetano Borriello. Real-world implementation of the Location Stack: The Universal Location Framework. In *Proceedings 5th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2003)*, pages 122–128. IEEE Computer Society Press, October 2003.

[GM02] Jean-Claude Geffroy and Gilles Motet. *Design of Dependable Computing Systems*. Kluwer Academic Publishers, 2002.

[Gon01] Daniel Jorge Viegas Gonçalves. Ubiquitous computing and AI towards an inclusive society. In *WUAUC'01: Proceedings of the 2001 EC/NSF Workshop on Universal Accessibility of Ubiquitous Computing*, pages 37–40. ACM Press, 2001.

[GS97] Rachid Guerraoui and André Schiper. Genuine atomic multicast. In *Proceedings of the 11th International Workshop on Distributed Algorithms (WDAG97)*, number 1320 in Lecture Notes in Computer Science, pages 141–154. Springer-Verlag, September 1997.

[GSB02] Hans-Werner Gellersen, Albrecht Schmidt, and Michael Beigl. Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts. *Mobile Networks and Applications*, 7(5):341–351, 2002.

[GSI05] Lee Gilbert, Sunanda Sangwan, and Mei Ian. Beyond usability: the OoBE dynamics of mobile data services markets. *Personal Ubiquitous Computing*, 9(4):198–208, 2005.

[GSM06] GSM Association. GSM World – the website of the GSM Association. `www.gsmworld.com`, February 2006.

[Gut96] P. Gutmann. Secure Deletion of Data from Magnetic and Solid-State Memory. In *Proceedings 6th USENIX Security Symposium, San Jose, California, USA*, July 1996.

[Gut01] P. Gutmann. Data Remanence in Semiconductor Devices. In *Proceedings 10th USENIX Security Symposium, Washington, D.C., USA*, August 2001.

[HaAA00] G. Hoblos and M. Staroswiecki amd A. Aitouche. Optimal design of fault tolerant sensor networks. In *Proceedings 2000 IEEE International Conference on Control Applications, Anchorage, AK, U.S.A.*, pages 467–472, September 2000.

[HB01] Jeffrey Hightower and Gaetano Borriello. Location Systems for Ubiquitous Computing. *IEEE Computer*, 34(8):57–66, August 2001.

[HBB02] Jeffrey Hightower, Barry Brumitt, and Gaetano Borriello. The Location Stack: A Layered Model for Location in Ubiquitous Computing. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002)*, pages 22–28, Callicoon, NY, June 2002. IEEE Computer Society Press.

[HBC+96] Thomas T. Hewett, Ronald Baecker, Stuart Card, Tom Carey, Jean Gasen, Marilyn Mantei, Gary Perlman, Gary Strong, and William Verplank. *Curricula for Human-Computer Interaction, Report of the ACM Special Interest Group on Computer-Human Interaction (SIGCHI) Curriculum Development Group*. Association for Computing Machinery, Inc., 1996. Available online at `http://www.sigchi.org/cdg/`.

[HBF+04] D. Hähnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose. Mapping and Localization with RFID Technology. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), New Orleans, LA, USA*, 2004.

[HC24] Jason L. Hill and David E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, November 22(6):12–24. November/December 2002.

[HCW04] Elgan Huang, Jon Crowcroft, and Ian Wassell. Rethinking incentives for mobile ad hoc networks. In *PINS '04: Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 191–196, New York, NY, USA, 2004. ACM Press.

[Hel00] A. Hellemans. Polymer matrix augurs flexible displays. *IEEE Spectrum*, 37(12):18–21, December 2000.

[HFG+98] Beverly L. Harrison, Kenneth P. Fishkin, Anuj Gujar, Carlos Mochon, and Roy Want. Squeeze me, hold me, tilt me! an exploration of manipulative user interfaces. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–24, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[HH04] Abdelsalam Helal and Joachim Hammer. UbiData: requirements and architecture for ubiquitous data access. *SIGMOD Rec.*, 33(4):71–76, 2004.

[HHS+02] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. *Wireless Networks*, 8(2/3):187–197, 2002.

[HHZK01] Sumi Helal, Joachim Hammer, Jinsuo Zhang, and Abhinav Khushraj. A Three-Tier Architecture for Ubiquitous Data Access. In *Proceedings ACS/IEEE International Conference on Computer Systems and Applications*, page 177. IEEE Computer Society, 2001.

[Hit06] Hitachi Ltd. Hitachi mu-chip - the World's smallest RFID IC. Hitachi Mu-Solutions. Homepage at `www.hitachi.co.jp/Prod/mu-chip/`, 2006.

[HKS04] Hung-Yun Hsieh, Kyu-Han Kim, and Raghupathy Sivakumar. An end-to-end approach for transparent mobility across heterogeneous wireless networks. *Mobile Networks and Applications*, 9(4):363–378, 2004.

[HKZ02] Abdelsalam Sumi Helal, Abhinav Khushraj, and Jinsuo Zhang. Incremental Hoarding and Reintegration in Mobile Environments. In *SAINT '02: Proceedings of the 2002 Symposium on Applications and the Internet*, pages 8–11, Washington, DC, USA, 2002. IEEE Computer Society.

[HL01] D. L. Hall and J. Llinas, editors. *Handbook of Multisensor Data Fusion*. CRC Press, 2001.

[HMKR04] M. Hollick, I. Martinovic, T. Krop, and I. Rimac. A survey on dependable routing in sensor networks, ad hoc networks, and cellular networks. In *Proceedings of 30th Euromicro Conference*, pages 495–502, 2004.

[HMNS01] Uwe Hansmann, Lothar Merk, Martin Nicklous, and Thomas Stober. *Pervasive Computing Handbook*. Springer, Heidelberg, 2001.

[HR83] Theo Härder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Computing Surveys*, 15(4):287–317, 1983.

[HS02] U. Hengartner and P. Steenkiste. Protecting People Location Information. UBICOMP Workshop on Security in Ubiquitous Computing, 2002.

[HT04] D. M. Hilbert and J. Trevor. Personalizing shared ubiquitous devices. *interactions*, 11(3):34–43, 2004.

[HZ04] A. Hohl and A. Zugenmaier. Safeguarding Personal Data with DRM in Pervasive Computing. In *Proceedings Security and Privacy Workshop at PERVASIVE 2004*. Kluwer Academic Publishing, 2004.

[IDA05] Infrared Data Association. Trade Association for Defining Infrared Standards, IrDA System Protocol consortium web site. `www.irda.org`, April 2005.

[IDE05] IDENTEC SOLUTIONS. ILR RFID Tag Family – i-Q Tags. `http://www.identecsolutions.com/i-q_tags.asp`, April 2005.

[IGE00] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 56–67, New York, NY, USA, 2000. ACM Press.

[inf04] informa telecoms & media. Mobile subscriber numbers exceed 1.5 billion. Analyst Report, 22. June 2004.

[inf05] informa telecoms & media. Worldwide Mobile Subscribers to Hit 2 Billions in 2005. Analyst Report, 7. February 2005.

[IoEEE90] Institute of Electrical and Electronics Engineers, editors. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. IEEE, New York, NY, U.S.A., 1990.

[IU97] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241. ACM Press, 1997.

[Jal94] P. Jalote. *Fault tolerance in distributed systems*. PTR Prentice Hall, Englewood Cliffs, N.J., USA, 1994.

[JEC05] Editorial Board JEC. Scope of Contents. *Journal of Embedded Computing*, 2005. Available online at `http://www.cisp-publishing.com/books/journalofembeddedcomputersscopeofcontents.htm`.

[JIMK03] Timo Jokela, Netta Iivari, Juha Matero, and Minna Karukka. The standard of user-centered design and the standard definition of usability: analyzing iso 13407 against iso 9241-11. In *CLIHC '03: Proceedings of the Latin American conference on Human-computer interaction*, pages 53–60, New York, NY, USA, 2003. ACM Press.

[JKH05] Arshad Jhumka, Stephan Klaus, and Sorin A. Huss. A dependability-driven system-level design approach for embedded systems. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 372–377, Washington, DC, USA, 2005. IEEE Computer Society.

[JW93] Niraj K. Jha and Sying-Jyan Wang. Design and synthesis of self-checking VLSI circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(6):878–887, June 1993.

[Kai01] Nikolaos Kaintantzis. Erweiterung von gedruckten Dokumenten um Online-Inhalte. Diplomarbeit, Institute for Information Systems, Department of Computer Science, ETH Zurich, Switzerland, April 2001.

[KB05] Frank Kargl and Alexander Bernauer. The COMPASS Location System. In *Proceedings 1st International Workshop on Location- and Context-Awareness (LoCA 2005), Oberpfaffenhofen, Germany*, number 3479 in LNCS, pages 105–112, May 2005.

[KBPD97] O. Kubitz, M. O. Berger, M. Perlick, and R. Dumoulin. Application of radio frequency identification devices to support navigation of autonomous mobile robots. In *IEEE 47th Vehicular Technology Conference*, volume 1, pages 126–130, May 1997.

[KCMT05] Todd Kort, Roberta Cozza, Kanae Maita, and Lillian Tay. PDA Market Has Record First Quarter, Growing 25 Percent. Gartner Report Nr. G00127580, 2. May 2005. Available online at `http://www.gartner.com/DisplayDocument?id=480706`.

[KFM03] C. Kwok, D. Fox, and M. Meila. Adaptive Real-Time Particle Filters for Robot Localization. In *Proceedings of the IEEE International Conference on Robotics & Automation*, 2003.

[KHM+00] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-Camera Multi-Person Tracking for EasyLiving. In *Proceedings 3rd IEEE Workshop on Visual Surveillance*, 2000.

[KHTK00] Sneha Kumar Kasera, Gísli Hjálmtýsson, Donald F. Towsley, and James F. Kurose. Scalable reliable multicast using multiple multicast channels. *IEEE/ACM Transactions on Networking*, 8(3):294–310, 2000.

[Kid99] Junji Kido. Organic displays. *Physics World*, 1999(3):27–30, March 1999.

[Kin02] Tim Kindberg. Implementing physical hyperlinks using ubiquitous identifier resolution. In *Proceedings 11th International Conference on World Wide Web*, pages 191–199. ACM Press, 2002.

[KL99] Steinar Kristoffersen and Fredrik Ljungberg. Designing Interaction Styles for a Mobile Use Context. In *Proceedings of 1st International Symposium Handheld and Ubiquitous Computing (HUC'99), Karlsruhe, Germany, September 27-29, 1999*, pages 281–288, 1999.

[Kni02] John C. Knight. Dependability of embedded systems. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 685–686, New York, NY, USA, 2002. ACM Press.

[KOB01] T. Kleine-Ostmann and A. E. Bell. A data fusion architecture for enhanced position estimation in wireless networks. *IEEE Communications Letters*, 5(8):343–345, 2001.

[Koh81] Walter H. Kohler. A survey of techniques for synchronization and recovery in decentralized computer systems. *ACM Computing Surveys*, 13(2):149–183, 1981.

[Kok04] M. M. Kokar. Situation awareness: issues and challenges. In *Proceedings 7th International Conference on Information Fusion*, pages 533–534, 2004.

[Kop04] Hermann Kopetz. An Integrated Architecture for Dependable Embedded Systems. In *SRDS '04: Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems (SRDS'04)*, pages 160–161, Washington, DC, USA, 2004. IEEE Computer Society.

[KP97] Geoffrey H. Kuenning and Gerald J. Popek. Automated hoarding for mobile computers. In *SOSP '97: Proceedings of the sixteenth ACM symposium on Operating systems principles*, pages 264–275, New York, NY, USA, 1997. ACM Press.

[KPS04] Aman Kansal, Dunny Potter, and Mani B. Srivastava. Performance aware tasking for environmentally powered sensor networks. In *SIGMETRICS 2004/PERFORMANCE 2004: Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 223–234, New York, NY, USA, 2004. ACM Press.

[KR99] U. Kubach and K. Rothermel. A universal, location-aware hoarding mechanism. *Proceedings of Handheld and Ubiquitous Computing*, 1707:377–379, 1999.

[KRA99] Naohiko Kohtake, Jun Rekimoto, and Yuichiro Anzai. InfoStick: An Interaction Device for Inter-Appliance Computing. In *Proceedings of 1st International Symposium Handheld and Ubiquitous Computing (HUC'99), Karlsruhe, Germany, September 27-29, 1999*, pages 246–258, 1999.

[KS92] James J. Kistler and Mahadev Satyanarayanan. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems*, 10(1):3–25, 1992.

[KWS02] John Krumm, Lyndsay Williams, and Greg Smith. SmartMoveX on a Graph - An Inexpensive Active Badge Tracker. In *Proceedings of the 4th international conference on Ubiquitous Computing (UbiComp '02)*, pages 299–307, London, UK, 2002. Springer.

[KWSB05] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello. Extracting places from traces of locations. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(3):58–68, 2005.

[Lal85] Parag K. Lala. *Fault tolerant and fault testable hardware design*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1985.

[Lam77] Leslie Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, 3(2):125–143, March 1977.

[Lam81] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, 1981.

[Lam83] L. Lamport. The Weak Byzantine Generals Problem. *Journal of the ACM*, 30(3):668–676, 1983.

[Lan01] Marc Langheinrich. Privacy by Design – Principles of Privacy-Aware Ubiquitous Systems. In *Proceedings Ubicomp 2001*, number 2201 in Lecture Notes in Computer Science, pages 273–291. Springer, 2001.

[Lan02] Marc Langheinrich. A Privacy Awareness System for Ubiquitous Computing Environments. In Gaetano Borriello and Lars Erik Holmquist, editors, *4th International Conference on Ubiquitous Computing (Ubicomp 2002)*, number 2498 in Lecture Notes in Computer Science, pages 237–245. Springer, September 2002.

[Lan05] Marc Langheinrich. *Privacy in Ubiquitous Computing*. Ph.d. thesis, no. 16100, ETH Zurich, Switzerland, 2005.

[Lap85] Jean-Claude Laprie. Dependable computing and fault tolerance: concepts and terminology. In *Proceedings of the 15th IEEE Symposium on Fault Tolerant Computing Systems (FTCS-15)*, pages 2–11, June 1985.

[Lap92a] Jean-Claude Laprie. Dependability: A Unifying Concept for Reliable, Safe, Secure Computing. *Proceedings of the IFIP Transactions, 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92*, pages 585–593, 1992.

[Lap92b] Jean-Claude Laprie, editor. *Dependability: Basic Concepts and Terminology*, volume 5 of *Dependable Computing and Fault-Tolerant Systems*. Springer, 1992.

[LC04] M.-H. Lin and C.-C. Chang. A secure one-time password authentication scheme with low-computation for mobile communications. *SIGOPS Operating Systems Review*, 38(2):76–84, 2004.

[LEG06] LEGO Mindstorms. Homepage at `http://mindstorms.lego.com`, 2006.

[Lev02] Nancy G. Leveson. An Approach to Designing Safe Embedded Software. In *EMSOFT '02: Proceedings of the Second International Conference on Embedded Software*, number 2491 in Lecture Notes in Computer Science, pages 15–29, London, UK, 2002. Springer.

[LF04] Matthias Lampe and Christian Flörkemeier. The Smart Box Application Model. In Alois Ferscha, Horst Hörtner, and Gabriele Kotsis, editors, *Advances in Pervasive Computing*, pages 351–356. Austrian Computer Society (OCG), April 2004.

[LH02] Minsoo Lee and Sumi Helal. HiCoMo: High Commit Mobile Transactions. *Distributed and Parallel Databases*, 11(1):73–92, 2002.

[LM98] Ulf Leonhardt and Jeff Magee. Multi-sensor location tracking. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 203–214, New York, NY, USA, 1998. ACM Press.

[LMFJ+04] Konrad Lorincz, David J. Malan, Thaddeus R. F. Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoffrey Mainland, Matt Welsh, and Steve Moulton. Sensor Networks for Emergency Response: Challenges and Opportunities. *IEEE Pervasive Computing*, 3(4):16–23, 2004.

[LMS85] Leslie Lamport and P. M. Melliar-Smith. Synchronizing Clocks in the Presence of Faults. *Journal of the ACM*, 32(1):52–78, January 1985.

[LRH00] Peter Ljungstrand, Johan Redström, and Lars Erik Holmquist. WebStickers: Using Physical Tokens to Access, Manage and Share Bookmarks to the Web, April 2000.

[LS81] Butler W. Lampson and Howard E. Sturgis. Atomic transactions. In B. W. Lampson, M. Paul, and H. Siegert, editors, *Distributed Systems-Architecture and Implementation, Lecture Notes in Computer Science 105, Springer, 1981, ed B. Lampson, with M. Paul and H. Siegert*, volume 105 of *Lecture Notes in Computer Science*, pages 246–265. Springer New York, 1981.

[LSM+03] Hilty L., Behrendt S., Binswanger M., Bruinink A., Erdmann L., Fröhlich J., Köhler A., Kuster N., Som C., and Würtenberger F. Das vorsorgeprinzip in der informationsgesellschaft. auswirkungen des pervasive computing auf gesundheit und umwelt. Technical Report TA 46/2003, Zentrums für Technologiefolgen-Abschätzung TA-SWISS, Switzerland, August 2003.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[Luc99] Robert W. Lucky. Everything will be connected to everything else. *Connections. IEEE Spectrum*, March 1999. Available online at `http://www.argreenhouse.com/papers/rlucky/spectrum/connect.shtml`.

[LvKSP02] M. M. Lankhorst, H. van Kranenburg, A. Salden, and A. J. H. Peddemors. Enabling technology for personalizing mobile services. In *Proceedings 35th Annual Hawaii International Conference on System Sciences (HICSS 2002)*, pages 1464–1471, January 2002.

[Lyo01] David Lyon. Facing the Future: Seeking Ethics for Everyday Surveillance. *Ethics and Information Technology*, 3(3):171–180, July 2001.

[Maa98] Henning Maass. Location-aware mobile applications based on directory services. *Mobile Networks and Applications*, 3(2):157–173, 1998.

[Mak01] Pantelis Makris. Accessibility of Ubiquitous Computing: Providing for the Elderly. In *Proceedings of the 2001 EC/NSF Workshop on Universal Accessibility of Ubiquitous Computing*, Alcácer do Sal, Portugal, May 2001. Available online at `http://virtual.inesc.pt/wuauc01/procs/pdfs/makris_final.pdf`.

[Mat01a] Friedemann Mattern. The Vision and Technical Foundations of Ubiquitous Computing. *Upgrade*, 2(5):2–6, October 2001.

[Mat01b] Friedemann Mattern. Ubiquitous Computing: Vision und technische Grundlagen. *INFORMATIK-INFORMATIQUE 5/2001 (joint issue with Novática and Upgrade)*, pages 4–7, October 2001.

[Mat03] Friedemann Mattern. Vom Verschwinden des Computers – Die Vision des Ubiquitous Computing. In Friedemann Mattern, editor, *Total vernetzt*, pages 1–41. Springer, April 2003.

[Mat04] Friedemann Mattern. Wireless Future: Ubiquitous Computing. In *Proceedings of Wireless Congress 2004*, Munich, Germany, November 2004.

[Mat05] Friedemann Mattern. Die technische Basis für das Internet der Dinge. In Elgar Fleisch and Friedemann Mattern, editors, *Das Internet der Dinge – Ubiquitous Computing und RFID in der Praxis*. Springer, 2005.

[Maz03] Thomas Mazhuancherry. Fehlertolerante Dienstinfrastruktur durch Ausnutzung lokaler Redundanz in Ubiquitous-Computing-Umgebungen. Diplomarbeit, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, August 2003.

[MB03] B. A. Myers and M. Beigl. Handheld computing. *IEEE Computer*, 36(9):27–29, September 2003.

[MFJW+04] David Malan, Thaddeus Fulford-Jones, Matt Welsh, , and Steve Moulton. CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care. In *International Workshop on Wearable and Implantable Body Sensor Networks*, Imperial College London, United Kingdom, April 2004.

[MRK+03] Milan Milenkovic, Scott H. Robinson, Rob C. Knauerhase, David Barkai, Sharad Garg, Vijay Tewari, Todd A. Anderson, and Mic Bowman. Toward Internet Distributed Computing . *IEEE Computer*, 36(5):38–46, May 2003.

[MS92] Shivakant Mishra and Richard D. Schlichting. Abstractions for constructing dependable distributed systems. Technical Report TR 92-19, University of Arizona, U.S.A., August 1992.

[MS03] Friedemann Mattern and Peter Sturm. From Distributed Systems to Ubiquitous Computing – The State of the Art, Trends, and Prospects of Future Networked Systems. In Klaus Irmscher and Klaus-Peter Fähnrich, editors, *Proceedings KIVS 2003*, pages 3–25, Springer, February 2003.

[MS04] Roy A. Maxion and Daniel P. Siewiorek, editors. *Workshop on Human Computer Interaction and Dependability, 46th IFIP WG 10.4 Meeting*, Siena, Italy, July 2004. Available online at `http://www.laas.fr/IFIPWG/Workshops\&Meetings/46/index.htm`.

[MZSMM03] Diana Marculescu, Nicholas H. Zamora, Phillip Stanley-Marbell, and Radu Marculescu. Fault-Tolerant Techniques for Ambient Intelligent Distributed Systems. In *ICCAD '03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 348, Washington, DC, USA, 2003. IEEE Computer Society.

[NCN98] Klara Nahrstedt, Hao Chu, and Srinivas Narayan. QoS-Aware Resource Management for Distributed Multimedia Applications. *High-Speed Networking, Special Issue on Multimedia Networking*, 7(3/4):227–255, 1998.

[Nel98] G. J. Nelson. *Context-Aware and Location Systems*. PhD thesis, Clare College, University of Cambridge, UK, January 1998.

[Net96] Netscape Communications. Secure Sockets Layer (SSL) 3.0 Specification, November 1996.

[NLLP03] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil. LANDMARC: indoor location sensing using active RFID. In *Proceedings 1st IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, pages 407–415, March 2003.

[NLLP04] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. LANDMARC: indoor location sensing using active RFID. *Wireless Networks*, 10(6):701–710, 2004.

[NMH+02] Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld, and Mathilde Pignol. Generating remote control interfaces for complex appliances. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 161–170, New York, NY, USA, 2002. ACM Press.

[Noy04] Natalya F. Noy. Semantic integration: a survey of ontology-based approaches. *SIGMOD Rec.*, 33(4):65–70, 2004.

[NSN+97] Brian D. Noble, Mahadev Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, and Kevin R. Walker. Agile application-aware adaptation for mobility. In *SOSP '97: Proceedings of the sixteenth ACM symposium on Operating systems principles*, pages 276–287, New York, NY, USA, 1997. ACM Press.

[NTT03] NTT Corporation and Waseda University. Market Requirements for IrBurst, September 2003. Available online at `http://www.irda.org/associations/2494/files/Publications/IrBurst_MRD.doc`.

[OA00] R. J. Orr and G. D. Abowd. The Smart Floor: A mechanism for natural user identification and tracking. In *Proceedings SIGCHI Conference on Human Factors in Computing Systems, The Hague, Netherlands*, April 2000.

[OMG01] Object Managment Group OMG. Super Distributed Objects (SDO) Whitepaper. OMG Document sdo/01-07-05, July 2001. Available online at `http://www.omg.org/`.

[OMG04] Object Managment Group OMG. Platform Independent Model (PIM) & Platform Specific Model (PSM) for Super Distributed Objects (SDO), Version 1.0. OMG Document sdo/04-11-01, November 2004. Available online at `http://www.omg.org/`.

[OMG05] Object Management Group OMG. Homepage. `www.omg.org`, 2005.

[Ope06] Open Mobile Alliance. Technical Section. Material from Affiliates. Wireless Application Protocol (WAP) – Downloads, February 2006. Available online at `http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html`.

[Opr05] Nicola Oprecht. Positioning and Object Tracking Based on Super-Distributed RFID Tag Infrastructures. Master's thesis, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, March 2005.

[PCB00] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proceedings of the Sixth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM)*, August 2000.

[Per98] Charles E. Perkins. Mobile Networking through Mobile IP. *IEEE Internet Computing*, 2(1):58–69, 1998.

[Pet02] D. Peterson. Implementing PDAs in a College Course: One Professor's Perspective. *Syllabus Magazine*, November 2002.

[Pfe05] Tom Pfeifer. Redundant positioning architecture. *Computer Communications*, 28(13):1575–1585, August 2005.

[PFF+03] M. Philipose, K. P. Fishkin, D. Fox, D. Hahnel, and W. Burgard. Mapping and Localization with RFID Technology. Technical Report IRS-TR-03-014, Intel Research, December 2003.

[PHD02] Thomas Phan, Lloyd Huang, and Chris Dulan. Challenge:: integrating mobile wireless devices into the computational grid. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 271–278, New York, NY, USA, 2002. ACM Press.

[Phi06] Philips Semiconductors. I-CODE – Smart Label Technology. Homepage at `http://www.semiconductors.philips.com/products/identification/icode/`, 2006.

[Pir04] Vito Piraino. A Middleware for Robust Self-Organizing Services Based on Highly Redundant RFID Tag Infrastructures. Master's thesis, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, December 2004.

[PK00] G. J. Pottie and W. J. Kaiser. Wireless Integrated Network Sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[PPZ99] T. Pfeifer and R. Popescu-Zeletin. A Modular Location-Aware Service and Application Platform. In *Proceedings of the 4th IEEE Symposium on Computers and Communications (ISCC '99)*, pages 137–148, Washington, DC, USA, 1999. IEEE Computer Society.

[PS03] Danat Pomeranets and Stephan Schneider. Erweiterung und Evaluierung eines Positionierungsdienstes. Semester thesis, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, October 2003.

[Pyt04] Emmanuel Python. Secure RFID-based Document Reconstruction. Semester thesis, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, July 2004.

[PZSA04] Radu Popescu-Zeletin, Stephan Steglich, and Stefan Arbanowski. Pervasive Communication - A Human-Centered Service Architecture. In *Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'04)*, pages 140–146, Washington, DC, USA, 2004. IEEE Computer Society.

[QLIM01] G. Qiang, Z. J. Liu, S. Ishihara, and T. Mizuno. Enhanced mobile Internet protocol based on IPv6 addressing scheme for third generation wireless network. *IEICE Transactions on Communications*, E84B(4):885–891, 2001.

[Qod06] Qode. It's in the code. `www.quode.com`, May 2006.

[RA00] Jun Rekimoto and Yuji Ayatsuka. CyberCode: Designing augmented reality environments with visual tags. In *DARE '00: Proceedings of DARE 2000 on Designing augmented reality environments*, pages 1–10. ACM Press, 2000.

[RABB07] Julian Randall, Oliver Amft, Jürgen Bohn, and Martin Burri. LuxTrace – indoor positioning using building illumination. *Personal and Ubiquitous Computing*, 2007. In Press. Online version available at `http://dx.doi.org/10.1007/s00779-006-0097-0`.

[RAdS+00] J. M. Rabaey, M. J. Ammer, J. L. da Silva, D. Patel, and S. Roundy. PicoRadio supports ad hoc ultra-low power wireless networking. *Computer*, 33(7):42–48, 2000.

[RAKO03] Jun Rekimoto, Yuji Ayatsuka, Michimune Kohno, and Haruo Oba. Proximal interactions: A direct manipulation technique for wireless networking. In *Proceedings of INTERACT '03: IFIP TC13 International Conference on Human-Computer Interaction, September 2003, Zurich, Switzerland*, 2003.

[RAMC+04] Anand Ranganathan, Jalal Al-Muhtadi, Shiva Chetan, Roy Campbell, and M. Dennis Mickunas. MiddleWhere: A Middleware for Location Awareness in Ubiquitous Computing Applications. In *ACM/IFIP/USENIX 5'th International Middleware Conference*, Toronto, Ontario, Canada, October 2004.

[RAT05] J. Randall, O. Amft, and G. Tröster. Towards LuxTrace: Using solar cells to measure distance indoors. In *Proceedings of the 1st International Workshop on Location- and Context-Awareness (LoCA 2005 Springer LNCS) in cooperation with Pervasive 2005*, pages 40–51, Oberpfaffenhofen near Munich, Germany, May 2005. ISBN 3-540-25896-5.

[RB03] Michael Rohs and Jürgen Bohn. Entry Points into a Smart Campus Environment – Overview of the ETHOC System. In *Proceedings of the 23rd International Conference on Distributed Computing Systems – 3rd International Workshop on Smart Appliances and Wearable Computing (IWSAWC 2003)*, Providence, Rhode Island, USA, May 2003.

[Ree83] David P. Reed. Implementing atomic actions on decentralized data. *ACM Transactions on Computer Systems*, 1(1):3–23, 1983.

[Rek97] Jun Rekimoto. Pick-and-drop: A direct manipulation technique for multiple computer environments. In *ACM Symposium on User Interface Software and Technology*, pages 31–39, 1997.

[RFI03] RFID Journal. `www.rfidjournal.com`, 2003.

[RJH02] Gruia-Catalin Roman, Christine Julien, and Qingfend Huang. Network abstractions for context-aware mobile computing. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 363–373, New York, NY, USA, 2002. ACM Press.

[Roh04] Michael Rohs. Real-World Interaction with Camera-Phones. In *2nd International Symposium on Ubiquitous Computing Systems (UCS 2004)*, pages 39–48, Tokyo, Japan, November 2004.

[Rom01] Alexander Romanovsky. Coordinated atomic actions: how to remain ACID in the modern world. *ACM SIGSOFT Software Engineering Notes*, 26(2):66–68, 2001.

[Rom04] Tony Romero. Managing high availability systems. Embedded Computing Design: Special Feature. An OpenSystems Publishing, LLC publication, 2004. Available online at `http://www.embedded-computing.com/articles/romero/`.

[RS00] W. Rungsarityotin and T. Starner. Finding Location Using Omnidirectional Video on a Wearable Computing Platform. In *Proceedings of IEEE International Symposium on Wearable Computing (ISWC 2000)*, pages 61–68, 2000.

[RSFWH98] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[RSMD04] Kay Römer, Thomas Schoch, Friedemann Mattern, and Thomas Dübendorfer. Smart Identification Frameworks for Ubiquitous Computing Applications. *Wireless Networks*, 10(6):689–700, December 2004.

[Rus01a] John Rushby. A Comparison of Bus Architectures for Safety-Critical Embedded Systems. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, September 2001. Available online at `http://www.csl.sri.com/~rushby/abstracts/buscompare`. This is a long version of Bus Architectures for Safety-Critical Embedded Systems [Rus01b].

[Rus01b] John M. Rushby. Bus Architectures for Safety-Critical Embedded Systems. In *EMSOFT '01: Proceedings of the First International Workshop on Embedded Software*, pages 306–323, London, UK, 2001. Springer.

[RV03] Daler Rakhmatov and Sarma Vrudhula. Energy management for battery-powered embedded systems. *Transactions on Embedded Computing Systems*, 2(3):277–324, 2003.

[Sar01] Sanjay E. Sarma. Towards the Five-Cent Tag. Technical Report MIT-AUTOID-WH-006, MIT Auto-ID Center, 2001. www.autoidcenter.org/research/MIT-AUTOID-WH-006.pdf.

[Sat96] Mahadev Satyanarayanan. Fundamental challenges in mobile computing. In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC '96)*, pages 1–7, New York, NY, USA, 1996. ACM Press.

[Sat01] Mahadev Satyanarayanan. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, 8(4):10–17, 2001.

[SAW94] Bill N. Schilit, Norman I. Adams, and Roy Want. Context-Aware Computing Applications. In *Proceedings of the 1st Workshop on Mobile Computing Systems and Applications (WMCSA), Santa Cruz, CA, USA*, pages 85–90. IEEE Computer Society., December 1994.

[SBG99] Albrecht Schmidt, Michael Beigl, and Hans-W. Gellersen. There is more to context than location. *Computers and Graphics*, 23(6):893–901, 1999.

[SBS91] André Schiper, Kenneth Birman, and Pat Stephenson. Lightweight causal and atomic group multicast. *ACM Transactions on Computer Systems*, 9(3):272–314, 1991.

[Sch02] C. Schär. Heuristiken zur Positionsbestimmung in Gebäuden mittels Sensor-Fusion und 2D-Kartenmodell. Diplomarbeit, Institute for Information Systems, Department of Computer Science, ETH Zurich, Switzerland, August 2002.

[Sch03] Simon Schlachter. Learning Movement Patterns and Predicting Positions in a Grid-based Probabilistic Positioning System. Semester thesis, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, October 2003.

[SCI+01] Eugene Shih, Seong-Hwan Cho, Nathan Ickes, Rex Min, Amit Sinha, Alice Wang, and Anantha Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 272–287. ACM Press, 2001.

[SDHM03] Marc Smith, Duncan Davenport, Howard Hwa, and Lik Mui. The Annotated Planet: A mobile platform for object and location annotation. In *Proceedings 1st International Workshop on Ubiquitous Systems for Supporting Social Interaction and Face-to-Face Communication in Public Spaces at UbiComp 2003, Seattle, Washington, USA*. Microsoft Research, October 2003.

[SET05] SETI@home. Search for Extraterrestrial Intelligence (SETI). Home page. `http://setiweb.ssl.berkeley.edu/`, 2005.

[SFV04] Frank Siegemund, Christian Flörkemeier, and Harald Vogt. The Value of Handhelds in Smart Environments. In Christian Mueller-Schloer, Theo Ungerer, and Bernhard Bauer, editors, *17th International Conference on Architecture of Computing Systems - Organic and Pervasive Computing (ARCS 2004)*, number 2981 in Lecture Notes in Computer Science, pages 291–308, Augsburg, Germany, March 2004. Springer.

[SHLX03] Arnaud Sahuguet, Richard Hull, Daniel F. Lieuwen, and Ming Xiong. Enter once, share everywhere: User profile management in converged networks. In *Proceedings 1st Biennial Conf. on Innovative Data Systems Research (CIDR 2003)*, Asilomar, CA, USA, January 2003.

[Sie04a] Frank Siegemund. A Context-Aware Communication Platform for Smart Objects. In Alois Ferscha and Friedemann Mattern, editors, *Pervasive Computing: Second International Conference, PERVASIVE 2004*, number 3001 in Lecture Notes in Computer Science, pages 69–86, Linz/Vienna, Austria, April 2004. Springer. (c) Springer.

[Sie04b] Frank Siegemund. *Cooperating Smart Everyday Objects – Exploiting Heterogeneity and Pervasiveness in Smart Environments*. PhD thesis, ETH Zurich, Zurich, Switzerland, December 2004.

[Sil97] Barry G. Silverman. Computer Reminders and Alerts. *Computer*, 30(1):42–49, 1997.

[SK04] Frank Siegemund and Pascal Keller. Tuplespace-based collaboration for bluetooth-enabled devices in smart environments. In *Proceedings Informatik 2004, 34. Jahrestagung der Gesellschaft fuer Informatik, 2nd German Workshop on Mobile Ad Hoc Networks*, Ulm, Germany, September 2004.

[SLP05] Thomas Strang and Claudia Linnhoff-Popien, editors. *Proceedings of the 1st International Workshop on Location- and Context-Awareness (LoCA 2005), Oberpfaffenhofen, Germany*, volume 3479 of *Lecture Notes in Computer Science*. Springer, May 2005.

[SMPC02] Z. Sahinoglu, F. Matsubara, K. A. Peker, and J. Cukier. A mobile network architecture with personalized instant information access. In *Digest of Technical Papers. International Conference on Consumer Electronics (ICCE 2002)*, pages 34–35, June 2002.

[SNB+01] E. Soloway, C. Norris, P. Blumenfeld, B. Fishman, J. Krajcik, and R. Marx. Log on education: Handheld devices are ready-at-hand. *Communications of the ACM*, 44(6):15–20, 2001.

[Son98] Y. Sonnenblick. An Indoor Navigation System for Blind Individuals. In CSUN Center On Disabilities, editor, *CSUN 1998 Conference, California State University Northridge*, Los Angeles, March 1998.

[SR03] Frank Siegemund and Michael Rohs. Rendezvous Layer Protocols for Bluetooth-Enabled Smart Devices. *Personal and Ubiquitous Computing*, pages 91–101, October 2003.

[SRS03] Curt Schurgers, Vijay Raghunathan, and Mani B. Srivastava. Power management for energy-aware communication systems. *ACM Transactions on Embedded Computing Systems*, 2(3):431–447, 2003.

[SS83] Richard D. Schlichting and Fred B. Schneider. Fail-stop processors: an approach to designing fault-tolerant computing systems. *ACM Transactions on Computer Systems*, 1(3):222–238, 1983.

[SSJ01] C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo. Sensor information networking architecture and applications. *IEEE Personal Communications*, 8(4):52–59, August 2001.

[SSR+99] Bernt Schiele, Thad Starner, Brad Rhodes, Brian Clarkson, and Alex Pentland. Situation Aware Computing with Wearable Computers. In W. Barfield and T. Caudell, editors, *Augmented Reality and Wearable Computers*. Lawrence Erlbaum Press, 1999.

[Ste01] Constantine Stephanidis. Towards Universal Access in the Information Society. Proceedings of the 2001 EC/NSF Workshop on Universal Accessibility of Ubiquitous Computing, May 2001. Available online at `http://virtual.inesc.pt/wuauc01/procs/pdfs/stephanidis_final.pdf`.

[Stu04] Lukas Stucki. System zur augenblicklichen Personalisierung und vorübergehenden Nutzung von mobilen Benutzergeräten. Semester thesis, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, July 2004.

[SVG+03] S. Steglich, R. N. Vaidya, O. Gimpeliovskaja, S. Arbanowski, R. Popescu-Zeletin, S. Sameshima, and K. Kawano. I-centric services based on super distributed objects. In *Proceedings 6th International Symposium on Autonomous Decentralized Systems (ISADS)*, pages 232–239, April 2003.

[SY85] Rob Strom and Shaula Yemini. Optimistic recovery in distributed systems. *ACM Transactions on Computer Systems*, 3(3):204–226, 1985.

[Tec04] Technology Working Group. Headed by A. Cuomo and L. Cloetens. Technology challenges for future Intelligent Embedded Systems. In *European Technology Platform on Embedded Systems Workshop, June 28–29, Rome, Italy*, June 2004.

[Tel00] Gerard Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, Cambridge, Eng.; New York, 2nd edition, 2000.

[Tim02] Tim Kindberg, John Barton, et al. People, Places, Things: Web Presence for the Real World. *Mobile Networks and Applications*, 7(5):365–376, 2002.

[TLAC95] Carl Tait, Hui Lei, Swarup Acharya, and Henry Chang. Intelligent file hoarding for mobile computers. In *MobiCom '95: Proceedings of the 1st annual international conference on Mobile computing and networking*, pages 119–125, New York, NY, USA, 1995. ACM Press.

[Tru03] Trusted Computing Group (TCG). TCG TPM Specification Version 1.2 (Revision 62), October 2003.

[TZVM04] P. Trakadas, Th. Zahariadis, S. Voliotis, and Ch. Manasis. Efficient routing in PAN and sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(1):10–17, 2004.

[UI00] B. Ullmer and H. Ishii. Emerging frameworks for tangible user interfaces. *IBM Systems Journal*, 39(3-4):915–931, 2000.

[UMT06] UMTS Forum. UMTS Forum Home. `www.umts-forum.org`, February 2006.

[VMKA02] E. Vildjiounaite, E. J. Malm, J. Kaartinen, and P. Alahuhta. Location estimation indoors by means of small computing power devices, accelerometers, magnetic sensors, and map knowledge. In *Pervasive '02: Proceedings of the 1st International Conference on Pervasive Computing*, pages 211–224. Springer, 2002.

[Vor05] Vorwerk & Co. Teppichwerke GmbH & Co. KG. Vorwerk is presenting the first carpet containing integrated RFID technology. Press release, Hamlin, Germany, June 2005.

[VTB04] Abhinav Vora, Zahir Tari, and Peter Bertok. An hoarding approach for supporting disconnected write operations in mobile environments. In *SRDS '04: Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems (SRDS'04)*, pages 276–288, Washington, DC, USA, 2004. IEEE Computer Society.

[VWG+03] M. Vossiek, L. Wiebking, P. Gulden, J. Weighardt, and C. Hoffmann. Wireless local positioning - concepts, solutions, applications. In *Radio and Wireless Conference*, 2003. RAWCON '03.

[WB97] Mark Weiser and John Seely Brown. The coming age of calm technolgy. In Peter J. Denning and Robert M. Metcalfe, editors, *Beyond calculation: the next fifty years*, pages 75–85. Copernicus, New York, NY, USA, 1997.

[WBB02] Andreas Weissel, Björn Beutel, and Frank Bellosa. Cooperative I/O: a novel I/O semantics for energy-aware applications. *ACM SIGOPS Operating Systems Review*, 36(SI):117–129, 2002.

[Wei91] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):66–75, September 1991. Reprinted in IEEE Pervasive Computing, 1(1), Jan.-Mar. 2002, pp. 19–25.

[Wei93a] Mark Weiser. Some Computer Science Problems in Ubiquitous Computing. *Communications of the ACM*, 36(7):75–84, July 1993.

[Wei93b] Mark Weiser. Ubiquitous Computing. *IEEE Computer*, 26(10):71–72, October 1993.

[Wei98] Mark Weiser. The Future of Ubiquitous Computing on Campus. *Communications of the ACM*, 41(1):41–42, January 1998.

[WFA06] Wi-Fi Alliance. Home Page. `www.wi-fi.org`, February 2006.

[WFGH99] Roy Want, Kenneth P. Fishkin, Anuj Gujar, and Beverly L. Harrison. Bridging physical and virtual worlds with electronic tags. In *Proceedings SIGCHI Conference on Human Factors in Computing Systems*, pages 370–377. ACM Press, 1999.

[WH04] Eric Hsiao-Kuang Wu and Yi-Zhan Huang. Dynamic adaptive routing for a heterogeneous wireless network. *Mobile Networks and Applications*, 9(3):219–233, 2004.

[WHFG92] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1), January 1992.

[Win99] Langdon Winner. The Voluntary Complexity Movement. *NETFUTURE: Technology and Human Responsibility*, September 1999. Available online at `http://www.oreilly.com/~stevet/netfuture/1999/Sep1499_94.html#3`.

[WJH97] A. Ward, A. Jones, and A. Hopper. A New Location Technique for the Active Office. *IEEE Personal Communications*, 4(5):42–47, October 1997.

[WOW+05] A. Wadaa, S. Olariu, L. Wilson, M. Eltoweissy, and K. Jones. Training a wireless sensor network. *Mobile Networks and Applications*, 10(1-2):151–168, 2005.

[WPD+02] R. Want, T. Pering, G. Danneels, M. Kumar, M. Sundar, and J. Light. The Personal Server: Changing the Way We Think about Ubiquitous Computing. In *Proceedings UbiComp 2002*, pages 194–209. Springer, 2002.

[WSA+97] Roy Want, Bill N. Schilit, Norman I. Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis, and Mark Weiser. *The ParcTab Ubiquitous Computing Experiment*, chapter 2, pages 45–101. In: Mobile Computing, Kluwer Publishing, February 1997.

[WVG04] Norbert Weißenberg, Agnès Voisard, and Rüdiger Gartmann. Using ontologies in personalized mobile applications. In *GIS '04: Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 2–11, New York, NY, USA, 2004. ACM Press.

[Xal02] XALAN Project Home Page. `http://xml.apache.org/xalan-j`, 2002.

[Xer02] XERCES Project Home Page. `http://xml.apache.org/xerces2-j`, 2002.

[YF00] B. Yao and W. K. Fuchs. Proxy-Based Recovery for Applications on Wireless Hand-Held Devices. In *Proceedings of 19th IEEE Symposium on Reliable Distributed Systems (SRDS 2000), Nürnberg, Germany*, pages 2–10, October 2000.

[YF01] B. Yao and W. K. Fuchs. Recovery Proxy for Wireless Applications. In *Proceedings of the 12th International Symposium on Software Reliability Engineering (ISSRE 2001)*, pages 112–119, November 2001.

[YHAP02] Kho Hao Yuan, Ang Chip Hong, M. Ang, and Goi Sio Peng. Unmanned library: an intelligent robotic books retrieval & return system utilizing RFID tags. In *Proceedings IEEE International Conference on Systems, Man and Cybernetics (SMC '02)*, volume 4, October 2002.

[ZHH03] J. Zhang, A. Helal, and J. Hammer. UbiData: ubiquitous mobile file service. In *Proceedings 2003 ACM Symposium on Applied Computing*, pages 893–900. ACM Press, March 2003.

[Zig06] ZigBee Alliance. ZigBee – Wireless Control That Simply Works. Home Page. `www.zigbee.org`, February 2006.

[Zwe04] Thomas Zweifel. Development and Simulation of Position-Reckoning-Strategies for Super-Distributed RFID Tag Infrastructures. Semester thesis, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, February 2004.