UNIVERSITÄT
DES
SAARLANDES

Spatial Modeling of Activity and User Assistance in Instrumented Environments

Dissertation

zur Erlangung des Grades des

Doktors der Ingenieurswissenschaften (Dr.-Ing.)

der Naturwissenschaftlich-Technischen Fakultäten der

Universität des Saarlandes

vorgelegt von

Dipl.-Inform. Christoph Stahl

Saarbrücken,

den 5. November 2009

Dekan:

Prof. Dr. Joachim Weickert

Vorsitzender des Prüfungsausschusses:

Prof. Dr. Philipp Slusallek

Berichterstatter:

Prof. Dr. Dr. h.c. mult. Wolfgang Wahlster

Prof. Dr. Bernd Krieg-Brückner

Prof. Dr. Antonio Krüger

Akademischer Beisitzer:

Dr.-Ing. Jörg Baus

Tag des Kolloquiums:

2. Dezember 2009

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

____

Saarbrücken, 5. November 2009

Short Abstract

This dissertation presents a design method for proactive user assistance systems in instrumented environments. The method addresses typical design issues, such as the modeling of users' needs and the choice and placement of sensors and actuators for human-environment interaction. The design process is supported through the combination of a geometric environment model and a situational semantic activity model. The geometric model is used to visualize the spatial context, in which the activities that are to be supported take place. The activity model is derived from Activity Theory and hierarchically represents tasks and activities in their situational context. Both models are linked by an ontology and form a hybrid location model. To support the method, we implemented a map modeling toolkit that allows to geometrically represent built environments in 3-D, and to model their furnishing and instrumentation with sensors and actuators. Of particular importance was the development and integration of an ontology-based activity editor. Furthermore, the toolkit facilitates the development of navigational aid through a route finding algorithm. The work concludes with five use cases that describe how the method and modeling toolkit have been applied for the design and development of intelligent environments and navigational aid for indoor and outdoor spaces. It also highlights how Dual Reality settings have contributed to the simulation of the developed assistance systems.

Kurzzusammenfassung

Diese Doktorarbeit legt eine Designmethode für proaktive Benutzerassistenzsysteme in instrumentierten Umgebungen vor. Die Methode richtet sich an typische Designprobleme, wie das Modellieren der Bedürfnisse der Benutzer sowie Auswahl und Platzierung von Sensoren und Aktuatoren zur Interaktion mit der Umgebung. Der Designprozess wird durch ein geometrisches Umgebungsmodell und ein situiertes, semantisches Aktivitätsmodell unterstützt. Das geometrische Modell wird verwendet, um den räumlichen Zusammenhang zu veranschaulichen, in dem die zu unterstützenden Tätigkeiten stattfinden. Das Aktivitätsmodell wird aus der Aktivitätstheorie abgeleitet und repräsentiert Tätigkeiten in ihrem situativen Kontext. Beide Modelle werden durch eine Ontologie verbunden und bilden ein hybrides Ortsmodell. Zur Unterstützung der Methode haben wir ein Modellierungswerkzeug implementiert, das es ermöglicht, Gebäudeumgebungen dreidimensional zu repräsentieren, und deren Einrichtung und Instrumentierung mit Sensoren und Aktoren zu modellieren. Besonders wichtig war die Entwicklung und Integration eines ontologiebasierten Aktivitätseditors. Weiterhin erleichtert das Werkzeug die Entwicklung von Navigationsassistenten durch einen Wegsuchealgorithmus. Die Arbeit schließt mit fünf Anwendungsbeispielen, in denen die Methode und das Modellierungswerkzeug für das Design und die Entwicklung von intelligenten Umgebungen und Navigationshilfen angewandt worden ist. Es wird auch gezeigt, wie Dual Reality Umgebungen zur Simulation der Assistenzsysteme beigetragen haben.

Acknowledgements

I want to thank the DFG (Deutsche Forschungsgesellschaft) for funding my position in the Collaborative Research Center 378 “Resource-Adaptive Processes”, and especially the Transfer Unit 53 “Resource-Adaptive Navigation”, which made this research possible.

Most of all, I want to thank Prof. Wahlster for providing the chance to work on my PhD at his AI chair, and for finding the time for intense supervision and inspiring discussions during the final stages of this work. Likewise, Prof. Bernd Krieg-Brückner contributed to this work by insightful discussions on AAL-related topics.

This work is rooted in our AI group’s collaborative effort to instrument the lab with ubiquitous computing technology and the experience of being involved in the development of various user assistance systems. I want to thank Antonio Krüger for his foresight to transform the lab into an instrumented environment, and all my colleagues from the chair of Prof. Wahlster for their commitment to *SUPIE*: Especially Jörg Baus for being a helpful and humorous office mate for so many years; Tim Schwartz for the joint work on indoor positioning; Dominik Heckmann for constant moral support and his UbisWorld; Rainer Wasinger for the fruitful collaboration on m3i; Michael Schmitz for developing the presentation manager; Mira Spassova for the steerable projector; Gerrit Kahl for delicious desserts; Boris Brandherm for his geoDBNs and bearing me company in the lab at oddest times; and finally our admins and Doris for taking care. At DFKI, Michael Schneider and Alexander Kröner have been involved in the SmartKitchen. In Bremen, I have to thank Tim Laue for his support with SimRobot.

I am especially thankful to Ichiro Kobayashi for for the successful collaboration towards activity modeling during his stay at the AI lab. It has also been a pleasure to work with Stefan Münzer, who contributed the wayfinding user study.

Last but not least I want to thank all the students who worked hard to implement and realize my ideas. First of all, credits go to Jens Haupert, who continuously implemented *YAMAMOTO* during 2003-2007. Without his patience and dedication, this toolkit would not exist. Likewise, my thanks go to Sebastian Wassmuth, who contributed the PathFinder algorithm. Acknowledgments also to Pascal Recktenwald for the Activity Editor component, Oliver Fickert for his extensive engagement towards the task planner, and Felix Schöttle for realizing the visual navigational aid. I also have to thank Michael Schmidt, Ingo Ruth, Philipp Brendel, and Andreas Maier for their contributions to the spatial audio guide.

Thanks also to Alexander and Yasemin for being best friends in all the good and bad times.

Most importantly, thank you Dimitra, for your love, appreciation, and constant support of my work. Σ'αγαπώ πολύ!

Christoph Stahl in November 2009

Table of Contents

List of Figures

List of Tables

List of Acronyms

2R	Dual Reality
AAL	Ambient Assisted Living
ABC	Activity-Based Computing
ACD	Activity-Centered Design
ADL	Activities of Daily Living
BAALL	Bremen Ambient Assisted Living Lab
DFKI	Deutsches Forschungszentrum für Künstliche Intelligenz
GOAL	Geometry, Ontology, Activity ModeL
GPS	Global Positioning System
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HEI	Human-Environment Interaction
OUI	Organic User Interface
PDA	Personal Digital Assistant
PNA	Personal Navigation Assistance
RFID	Radio Frequency IDentification
SUPIE	Saarland University Pervasive Instrumented Environment
TUI	Tangible User Interface
UBIDOO	UBIquitous To-DO Organizer
URC	Universal Remote Console
URI	Universal Resource Identifier
URL	Universal Resource Locator
VISTO	Videos for SpaTial Orientation
VR	Virtual Reality
XML	eXtensible Markup Language
YAMAMOTO	Yet Another MAp MOdelling TOolkit
YML	Yamamoto Modeling Language

1 Introduction

This chapter motivates the research questions that are investigated in this thesis. They are based on our experiences with the design and development of user assistance in intelligent environments.

1.1 Introduction

In 1991, Marc Weiser at Xerox PARC was one of the first to foresee how the miniaturization of information technology will change our life through a new computing paradigm for which he coined the term Ubiquitous Computing: Instead of using a single, personal computer on the desktop, we will be using multiple computing devices that are situated everywhere.

Today this change is happening indeed, as the number of mobile devices in our lives increases, while time spent on PC's decreases. It is not unusual that people use two mobile phones, one for private and one for business calls, a pocket-sized digital camera, an additional SLR camera, and a personal navigation device (PNA).

Most of these devices also support wireless networking, and standards like URC promise the migration of single components into smart home environments; media can be shared between server and multiple rendering devices according to the user's situation. Home automation is also advancing: automatic lights as well as intelligent climate control systems for efficient heating and cooling can contribute to save ecologic resources.

Besides comfort, home automation and assistance will soon become a necessity for the elderly. Due to expected demographic changes in most western countries, there will be not enough young people to take care for the seniors in nursing homes in the future. Hence the health care sector hopes for technical solutions that will allow people in need of care to stay in their homes as long as possible. The goal of assisted living projects, such as the European and German Ambient Assisted Living (AAL) Joint Programmes[1][2] or MIT's Age Lab[3], is to instrument homes with assistance systems that support the well being and independent living of people with health risks in their home environment. These systems address the physical and psychological needs of elderly people and provide means for social interaction and the achievement of personal goals. Furthermore, AAL aims to provide care givers, such as relatives and professional nurses, up-to-date information about the patient at home. In the long term, research aims towards support for cognitive impaired users (e.g. mild dementia caused by Alzheimer disease) by reminding them of their activities of daily living,

[1] http://www.aal-europe.eu/
[2] http://www.aal-deutschland.de/
[3] http://web.mit.edu/agelab/projects_independant.shtml

like cooking or dressing.

Outside of our homes, environments such as shopping malls, airports and museums could also employ modern technologies to assist their users and customers. Intelligent environments are predicted to aid their users in pursuing their activities, such as wayfinding or shopping, through the situated presentation of personalized information on mobile devices or stationary information kiosks.

Summing up, the possible applications of user assistance systems in activities of daily living are manifold. Finally, Weiser (1991) assumed that we will use computing technology for our everyday tasks without consciously thinking about it:

> *"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." (Weiser, 1991:94)*

However, his vision of the computer as an 'invisible' tool still seems like utopia, as user interfaces today compete for our attention with flashy rather than calm design.

In his column in the interactions journal, psychologist and designer Don Norman (2007) stated that one of the biggest design challenges is the ever increasing complexity of everyday things:

> *"As our demands for services, functions and features grow, the complexity of our devices – cameras and cell phones, audio and television equipment, kitchen appliances and automobile dashboards - increases beyond reasonable comprehension." (Norman, 2007:46)*

Krüger supports this attitude by arguing that the number of user interfaces in our environment increases, whilst the cognitive resources of the user remain on the same level. He claims that new devices will only be considered as beneficial to the user if they are designed for a net resource release gain. In other words, the perceived saving of time and attention has to be bigger than the necessary effort to learn and operate a new interface. Likewise José Encarnação recently (AmI'08 conference keynote speech) identified the human barriers as one of the greatest challenges for the success of Ambient Intelligence, and in particular in the AAL domain where seniors are confronted with unfamiliar high-tech.

The strategy to overcome the gap between increasing technical possibilities and limited human cognitive resources is to develop intelligent user interfaces which understand the user's needs and adapt themselves to the user's situation.

Take smartphones, for example. They combine several applications and functions, like a calculator, camera, alarm clock and calendar, which can be quite useful in certain everyday situations. However, it is up to the user to know about these functions, where to find them and how to use them. Many users cannot handle the complexity of such user interfaces, and end up in frustration or embarrassment. Norman (1986) gives a formal model of human-computer interaction that approximates the process of performing and evaluating an action in seven stages of activity. The user establishes the *goal*, forms the *intention* and then specifies the action sequence. The *actions* are executed and the system state is perceived by the user. Finally, the results are interpreted and evaluated with respect to *goals and intentions*.

Delegation- or goal-oriented user interfaces like *SmartKom* (Wahlster et al., 2001) or *EMBASSI* (Herfet et al., 2001) are a better approach, where the system allows its users to directly formulate their goals and intentions instead of carrying out a sequence of operations. The system automatically plans the necessary actions and executes them.

By incorporating ubiquitous computing, so-called Instrumented (or Intelligent) Environments can sense the user's context, such as location and activity, in order to adapt themselves to the user's situation. Proactive systems go one step further and try to recognize the intentions of the users themselves through perceiving the user's actions which are not directly related to the system itself. The system interprets the user's goals and automatically plans and suggests corresponding actions to support the user. This concept is also known as *implicit interaction* (Schmidt, 2000).

We define proactive user assistance in this context as follows:

Definition: A **Proactive user assistance system for instrumented environments** is a system which proactively supports users in their activities by sensing contextual information to improve the presentation and interaction between them and the instrumented environment.

1.2 User Assistance in Instrumented Environments

In this section we will describe our research on the ubiquitous computing paradigm in the project REAL - Dialogue and Interaction in Instrumented Spaces. This project has been part of the Collaborative Research Center 378 "Resource-Adaptive Systems". In 1999, the second generation of *PalmPilot* Personal Digital Assistants, the *Palm III*, has been widely popular and allowed us to work with devices quite similar to Weiser's Parc tabs. We used them to implement a pedestrian navigation system that is based on infrared beacons for indoor positioning. In 2002, plasma displays similar to the Parc board became widely available, and RFID tags were truly the first 'invisible' computers to disappear in objects of everyday life. We set up an instrumented environment that consists of distributed computational power, presentation media, and sensors to observe and recognize implicit user interactions. This offered the possibility to infer about a user's plans and intentions and to proactively assist in solving their task, so we have focused our research on the following question:

How can a system assist its user in solving different tasks in an instrumented environment?

Though our research prototypes are restricted to our lab environment, our motivation stems from public environments where people are likely to benefit from assistance services. We focused our interest on two particular tasks in an airport scenario: i) navigation and ii) shopping, which are partially overlapping as illustrated in Figure 1. Imagine a user, who has just checked in at a big airport and wants to buy a digital camera at the duty-free zone before the boarding of the plane. The first goal is to find a shop which offers a good selection of digital cameras (1a). Facing the shelf with many digital cameras, the user tries to find a certain model, which has been recommended by a friend (1b). Next, the user compares the camera against sales offers (1c), before he finally has to find his way to the departure gate.

We developed three prototype systems, which proactively aid the user in each activity. In order to support navigational tasks on a macro-level (e.g. to enter the shop), we implemented a mobile PDA-based pedestrian navigation system, the *BMW Personal*

Navigator (Krüger et al., 2004), which receives RFID and infrared beacon information for indoor positioning and provides both graphical and spoken wayfinding directions to the user. On a finer scale, the *SearchLight* (Butz et al., 2004) system uses a steerable projector to highlight products in the shelf, which we consider as micro-navigational aid for object localization. Our third system, the *Smart Shop Assistant* (Schneider, 2004), employs an RFID-technology based infrastructure of readers and labeled products to sense implicit user interactions, such as picking up a product from the shelf or putting it into the shopping cart. It proactively provides the user with product details and also generates comparison information if more than one product is taken out of the shelf.

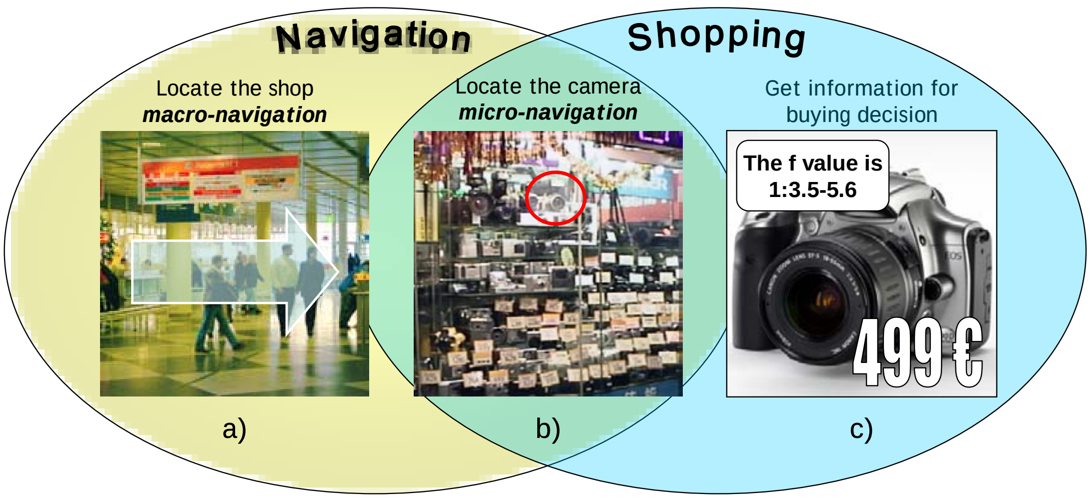


Figure 1: Overlapping activities of navigation and shopping in the airport scenario.

1.3 Lessons Learned on the Design of Intelligent Environments

The prototype systems of our instrumented environment have successfully demonstrated how ubiquitous computing technology can proactively provide user assistance in everyday life situations. However, we made the experience that designing, implementing, and evaluating user interfaces for instrumented environments requires new methods that accommodate the ubiquity of the systems. One of the first publications on design principles for intelligent environments is that by Coen (1998b), where he concludes:

> *"Intelligent Environments need to be more than rough assemblages of previously existing systems. In particular, careful selection and communication among modalities can lead to synergetic reinforcement and overall, a more reliable system. The modalities must also be carefully selected in order to make the environment easy to install, maintain, and use under a wide range of environmental conditions."(Coen, 1998b:553)*

Coen's statement addresses the integration of subsystems into intelligent environments. The intelligence of an environment does not automatically increase by simply adding technology. In fact, intelligence emerges from the sharing of information about the user between the different systems and their ability to adapt to the current situation of the user.

Despite the fact that the navigational and shopping tasks in our scenario are overlapping, our assistance systems have not been integrated into a seamless user interface that guides the user through the whole process. In order to buy a product, the user has to first select the respective shop as navigational goal on the PDA. Inside the shop, the user has to manually change from the navigation to the shopping application and select the product from a list in order to receive micro-navigational aid and product information. One reason is the task-oriented design of both systems; the navigation system assists in the wayfinding task, and the shop assistance system only provides product information. However, both systems are unaware of the higher-level traveling activity. This dilemma is quite representative and can be observed today e.g. in car navigation systems.

Coen (1998b) also points out the importance to carefully select and integrate multiple modalities for interaction with the intelligent environment. The combination of different input channels can help overcome the limitations of a single modality like speech. Speech utterances are often ambiguous in their meaning, and the system can only guess the speaker's intention. In combination with gestures, the system can disambiguate between several hypotheses and the error rate goes down.

The selection of modalities should also reflect the situation of the user, for example it is difficult to read text on the display of a mobile device while walking, so that audio output is more appropriate (Vadas et al., 2006). Input and output channels are also dependent on the environmental conditions. For example, changing light conditions during the day affects computer vision, background noise affects speech recognition, and metal or liquids absorb electromagnetic energy from RFID tags. Hence the interaction components should be chosen carefully considering their actual placement in the environment. We made the experience that especially the placement of visual elements like displays, projectors, and cameras is difficult to plan, since their field of view heavily depends on the building's architectural features and interior design.

The proposed arrangement of interaction devices and their physical placement needs to be documented and communicated to the development team. Each device has to be described in multiple facets: the physical location where it is located, its technical specification, the acquisition process, and the software configuration like network address and login. We felt that using traditional office applications to create and exchange sketches and spreadsheets is very limiting, and that specialized design tools could make the workflow more efficient.

Finally, not only the design and the specification of our instrumented environment, but also its practical realization turned out to be challenging. As such environments comprise many networked subsystems, tracing errors is a complicated procedure. If one observes a malfunction, it is not evident which component causes the problem. In our project, the debugging of the components has been difficult since their logfiles were spread across multiple Unix-based servers, and analyzing their content was cognitively demanding.

1.4 Subject and Goal of this Thesis

From the experiences that we gained in our AI group during the project REAL, which we summarized in the introduction, we believe that current software-development methods and tools are not sufficient to: i) *efficiently plan, realize, and debug assistance systems in instrumented environments*, and ii) *achieve the goal of a seamless integration of multiple services*.

The **goal of this thesis** is to investigate how the concepts of activity and location can be applied in order to aid the design of proactive user assistance systems in instrumented environments and furthermore to contribute both a new method and a toolkit that support the development of such systems.

Our approach is theoretically founded on a semantically-grounded activity model with situational context. We borrow the basic structure of the activity model from Activity Theory (Leontiev, 1978), that defines activity as purposeful interaction of a subject with the world. According to the theory, activities can be hierarchically structured in: i) activities, ii) actions, and iii) operations. The theory also comprises the principle of *mediation* that refers to the special role of culturally developed artifacts, such as tools or signs, in human activities. All together, this yields the following model of activity:

Definition: **Activity model.** According to Activity Theory (Leontiev, 1978), activity is hierarchically structured as activities, actions, and operations. Activities are directly motivated by the users' needs, whereas actions pursue certain goals towards these needs. Actions are decomposed into operations that can be carried out without consciousness thought. Artifacts represent tools or signs that mediate cultural knowledge or functionality. Based on these concepts, activity can be modeled as a tree of (*subject*, *artifact*, *object*) triples.

Starting with this definition of an activity model, we will investigate how ontologies can be used in order to embed the model into a semantic framework, and how to establish a relationship to the situational context of activities. We will now introduce and define the following concepts that will be used throughout this thesis:

Definition: A **semantic activity model** is defined as triple of (*subject*, *artifact*, *object*), whereas said subject, artifact, and object refer to elements of an ontology that defines their semantics.

Definition: A **situational semantic activity model** extends said activity triple of (*subject*, *artifact*, *object*) with references to the situational context in which the activity is typically performed, with respect to location, time, and human-environment interaction. Such a model can be conceived as a six-tuple of (*subject*, *artifact*, *object*, *location*, *time*, *instrumentation*).

Practically, this thesis aims to develop an integrated set of tools that support the design process through: i) the geometric modeling of the environment and ii) the editing of a situational semantic activity model with respect to the prior. In order to support the implementation, the debugging, and the evaluation of instrumented environments, we strive towards a tight coupling between the virtual environment model and the real world. This concept, also known as *Dual Reality*, is depicted in Figure 2 and assumes that a virtual model (left) of the real world exists (right) and both worlds are connected through a network, such as a home automation bus. User activity in the real world is represented in the virtual world, and the avatar can also control the real world.

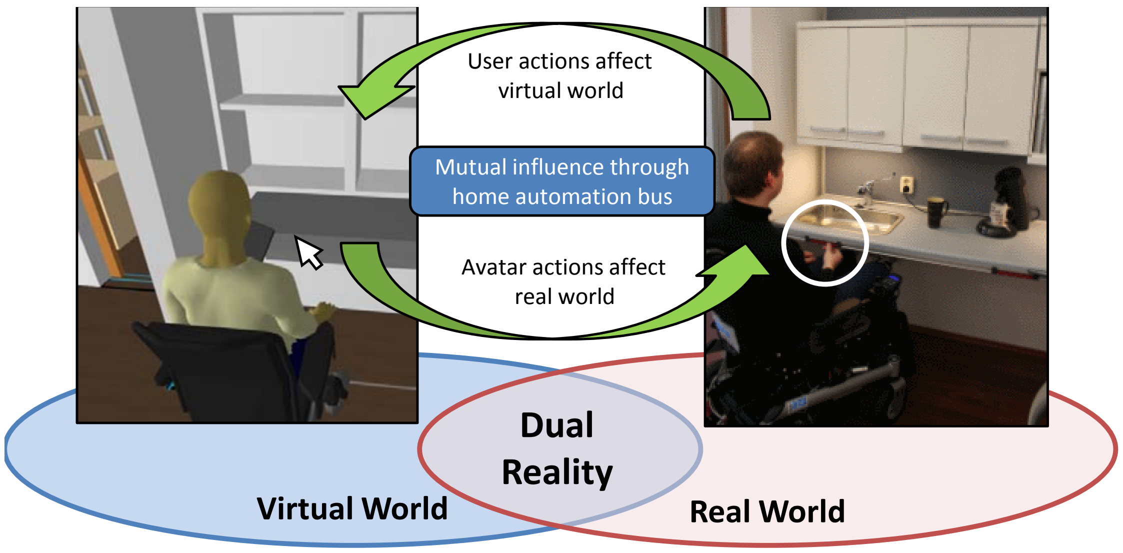

User actions affect virtual world
Mutual influence through home automation bus
Avatar actions affect real world
Dual Reality
Virtual World
Real World

Figure 2: The concept of Dual Reality.

Regarding the method and the tools, we will investigate the following research questions:

- **How to methodically approach the design and development of user assistance?**
  The complex nature of everyday activities and the ubiquitous computing technologies that are involved make the development of such systems a difficult venture. We will investigate how a situational semantic activity model can be methodically applied in order to model the users' needs with respect to the environmental context. The method should support all stages of the system development process.
- **How to aid the design and development of instrumented environments through software tools?**
  Our method will be based on a situational semantic activity model with spatial reference. We have to develop a tool that allows the geometric modeling and visualization of the environment in 3-D. We will investigate solutions for the special requirements of built environments, such as multiple levels, furnishing, and route finding.

The developed toolkit should be particularly helpful for the designer in order to answer the following critical design questions:

- **Which activities and tasks require assistance?**
  Clearly, not all activities of everyday life will benefit from high-tech gadgets. We will investigate how to systematically assess the user's needs based on an activity model.
- **How to map an activity model into interactions with computing artifacts?**
  Once we have figured out critical actions that require assistance, we have to define assistance systems to interact with.
- **Which modalities should be used for interaction, and where should the necessary technologies be deployed?**
  Whereas the field of human-computer interaction (HCI) has been traditionally concerned with the design and layout of components for graphical user interfaces on a single screen, we have now to design multimodal interfaces that comprise multiple sensors and actuators in different locations. I.e. we have to place displays and cameras considering their field of view, and, especially in the case of navigational aid, architectural context.

Regarding subsequent development phases, mostly practical issues arise:

- **How to evaluate a design concept in Virtual Reality?**
  Evaluating user assistance in real environments is time consuming and costly. We will investigate how the virtual environment model can be utilized to visualize and explore design concepts more rapidly.
- **How to debug intelligent environments in Dual Reality?**
  The complex relationship between sensor input and actuator output makes system failures very difficult to trace. Also, the remote debugging of environments is an important issue, considering travel expenses. We will investigate how the concept of *Dual Reality* can be applied to visualize and manipulate the state of the real world in a virtual environment.

1.5 Thesis Outline

This thesis consists of 9 chapters and their summaries are provided below. The *GOAL* design method and the *YAMAMOTO* toolkit are the main contributions.

Chapter 2 presents basic concepts of interaction design, and introduces terms and principles of Activity Theory terms as foundation for our new approach for activity-adaptive systems. Furthermore, we introduce concepts for wayfinding and navigation.

Chapter 3 relates this thesis to other projects in the field of instrumented environments, and we present related approaches for activity and location modeling.

Chapter 4 defines strategies to overcome the limitations of traditional function-oriented user interfaces: user-adaptive systems, goal-based interaction, and proactive assistance. In addition, we introduce our new concept of activity-based systems.

Chapter 5 gives a survey of our project REAL, in which we applied implicit interaction and ubiquitous computing technologies to provide proactive user assistance for shopping and navigation tasks. We also present the technical infrastructure.

Chapter 6 outlines the *GOAL* design method for user assistance in instrumented environments; we investigate how a geometric environment model and a situational semantic activity model can be applied to aid the design of instrumented environments.

Chapter 7 introduces the *YAMAMOTO* toolkit that supports the integrated modeling of location and activity and aids the designer of instrumented environments during the design process, as described in the previous chapter.

Chapter 8 presents five actual use cases for our method and toolkit. We focus on activity modeling in an instrumented kitchen environment, activity-adaptive navigational aid for pedestrians in buildings, and the physical simulation of an ambient assisted living lab with a robotic wheelchair and physical actuators. Furthermore, we describe navigational assistance in a supermarket, and an audio guide for the zoo.

Chapter 9 gives a summary and outlook.

2 Background & Basic Concepts

This chapter introduces basic concepts to explain what intelligent environments are, and the theoretical background that is necessary to understand our proposed method and tools for the modeling and design of user assistance in instrumented environments later in this thesis. We begin with the basic principles of interaction design, which comprise a summary of conceptual models like direct manipulation, and a survey of interaction paradigms, such as Ubiquitous Computing and Ambient Intelligence. Since our design approach is based on the modeling of activity and location, we also give a detailed introduction into the roots and principles of Leontiev's Activity Theory and motivate their possible application to interaction design according to authors like Kaptelinin and Nardi. Regarding location, we briefly sketch the fundamentals of wayfinding and spatial knowledge, since our application domain includes assistance systems for pedestrian navigation. The chapter concludes with an overview on location models for ubiquitous computing, considering both geometrical and semantic aspects of geographic knowledge.

2.1 Interaction Design

In this section, we will present the basic concepts of interaction design according to Preece, Rogers, and Sharp (2002). It is motivated by the advent of information appliances and mobile computing technologies, which make it for the first time possible to design interactive products that support all aspects of a person's everyday and working life. Mobile phones have already begun to heavily influence all areas of life – at home, on the move, at school, at leisure as well as at work – and enhance and extend the way people communicate, interact and work. Winograd (1997) describes interaction design in more general terms as "the design of spaces for human communication and interaction". These definitions subsume the topic of this thesis, which is to develop user assistance systems in intelligent environments, so the framework presented by Preece et al. is a good starting point for this chapter.

We will begin this section with a survey of conceptual models, and contribute an overview of interface metaphors and interaction paradigms, such as the aforementioned Ubiquitous Computing, that is oriented on the concepts of virtuality, reality, and mixed realities. Then we will introduce an iterative process model that includes prototyping and evaluation through real users, also known as participatory design or human-centered design.

2.1.1 Conceptual Models Based on Activities

According to Preece, Rogers, and Sharp (2002), a *conceptual model* is a description of the proposed system in terms of a set of integrated ideas and concepts about what it should do, behave and look like, that will be understandable by the users in the manner intended.

The conceptual model defines on a high level of abstraction what the users will be doing when they interact with the product and how they will be carrying out their tasks. By choosing the appropriate conceptual model for the user interface, the designer conveys how a system is implemented and intended to be used. It helps the user to develop a correct mental model of how the system works. Norman (1983) distinguishes between mental models and conceptual models:

> *"Conceptual models are devised as tools for the understanding or teaching of physical systems. Mental models are what people really have in their heads and what guides their use of things." (Norman, 1983:12)*

Norman further states that

> *"People form mental models through experience, training and instruction. The mental model of a device is formed largely by interpreting its perceived actions and its visible structure. I call the visible part of the device the system image. If the system image is inappropriate, the user will form a wrong mental model and cannot easily use the device. Whereas a correct mental model allows the user to reason about the system, to anticipate system behavior and to explain why the system reacts as it does." (Norman, 1988:17)*

In the following, we will discuss conceptual models that are based on activities, objects and metaphors.

The following conceptual models describe the way in which people interact with systems.

- **Instructing**. Users issue instructions in an imperative style to the system by typing commands or pressing buttons. The issued commands are carried out in sequence by the system, which returns the appropriate results. Examples for this type of systems are alarm clocks, vending machines *"insert coin - push button"*, the Unix shell, and desktop applications where the user picks his options with the mouse from graphical menus. Benefits of this model are quick and efficient interactions, and it is suitable for repetitive kinds of actions performed on multiple objects.
- **Conversing**. The system is designed to respond (similar to another human being) in a conversation and acts more like a partner than a machine. This model is most useful if the user needs to find out specific kinds of information in a dialogue that can't easily be selected through menus or buttons. Examples are search engines, help facilities and customer support systems in the style of QA dialogues.
- **Manipulating and navigating**. The user is manipulating virtual objects and navigating through virtual spaces. By selecting, moving, and clicking on graphical representations of virtual objects, the user performs various operations like opening a file, moving through folder hierarchies or zooming into an image. This concept exploits the user's knowledge of how to manipulate and navigate in the real word to operate with digital information. Examples include *what you see is what you get* (WYSIWYG) word processors, Spreadsheets like Microsoft Excel, and in general all kinds of video games. Shneiderman (1983; 1992:205) has formalized this concept as **direct manipulation** interfaces by three principles:
    - Continuous representation of the objects and actions of interest;

- o Physical actions or presses of labeled buttons instead of complex syntax;
- o Rapid incremental reversible operations whose effect on the object of interest is immediately visible.

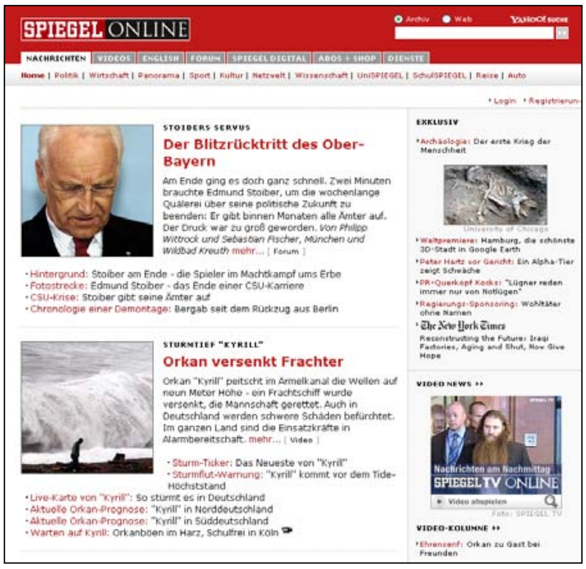
According to Shneiderman, using these principles leads to these benefits:

- o Users can immediately see if their actions are furthering their goals;
- o Users feel in control and experience less anxiety because the system is comprehensible and because actions can be reversed;
- o Error messages are rarely needed.

Whereas the principle of direct manipulation addresses the design of interfaces for the desktop computer, it is also important and applicable to devices in the real world, and Shneiderman introduces it by the example of driving a car; the scene is directly visible through the front window, and actions such as steering or braking provide immediate feedback to the driver as the scene changes.

- **Exploring and browsing**. This model is based on the idea of allowing people to explore and browse information, exploiting their knowledge of how to do this with existing media. The users scan and flick through parts of the information, hoping to find something interesting to read. Examples are Web portals and e-commerce sites.
- **Implicit and proactive**. In contrast to the other models, the user does not explicitly interact with the system. The system rather observes what the user is doing and responds to actions that are not directly addressed to the system. Simple examples are doors that automatically open or lights that are turned on if someone passes by.

These concepts are not mutually exclusive, and they are usually combined and used together. For example, the web page of a online news magazine shown in Figure 3 is designed to let the user browse and explore the content similar to a printed magazine, but it also allows to instruct the system to search for a specific information by entering search terms in the text box which is located in the upper right corner of the page. Video clips allow the user to manipulate their size and to navigate through their content using a timeline.



STOIBERS SERVUS

Der Blitzrücktritt des Ober-Bayern

Am Ende ging es doch ganz schnell. Zwei Minuten brauchte Edmund Stoiber, um die wochenlange Quälerei über seine politische Zukunft zu beenden: Er gibt binnen Monaten alle Ämter auf. Der Druck war zu groß geworden. *Von Philipp Wittrock und Sebastian Fischer, München und Wildbad Kreuth* mehr... [Forum]

- Hintergrund: Stoiber am Ende - die Spieler im Machtkampf ums Erbe
- Fotostrecke: Edmund Stoiber - das Ende einer CSU-Karriere
- CSU-Krise: Stoiber gibt seine Ämter auf
- Chronologie einer Demontage: Bergab seit dem Rückzug aus Berlin

STURMTIEF "KYRILL"

Orkan versenkt Frachter

Orkan "Kyrill" peitscht im Ärmelkanal die Wellen auf neun Meter Höhe - ein Frachtschiff wurde versenkt, die Mannschaft gerettet. Auch in Deutschland werden schwere Schäden befürchtet. Im ganzen Land sind die Einsatzkräfte in Alarmbereitschaft. mehr... [Video]

- Sturm-Ticker: Das Neueste von "Kyrill"
- Sturmflut-Warnung: "Kyrill" kommt vor dem Tide-Höchststand
- Live-Karte von "Kyrill": So stürmt es in Deutschland
- Aktuelle Orkan-Prognose: "Kyrill" in Norddeutschland
- Aktuelle Orkan-Prognose: "Kyrill" in Süddeutschland
- Warten auf Kyrill: Orkanböen im Harz, Schulfrei in Köln

EXKLUSIV

- Archäologie: Der erste Krieg der Menschheit
- Weltpremiere: Hamburg, die schönste 3D-Stadt in Google Earth
- Peter Hartz vor Gericht: Ein Alpha-Tier zeigt Schwäche
- PR-Querkopf Kocks: "Lügner reden immer nur von Notlügen"
- Regierungs-Sponsoring: Wohltäter ohne Namen
- The New York Times
  Reconstructing the Future: Iraqi Factories, Aging and Shut, Now Give Hope

VIDEO NEWS ››

VIDEO-KOLUMNE ››

- Ehrensenf: Orkan zu Gast bei Freunden

Figure 3: The *Spiegel Online* user interface combines three activity-based concepts.

2.1.2 Conceptual Models Based on Objects and Metaphors

Conceptual models can also be based on real-world objects. The advantage of designing user interfaces in analogy to well-known objects is that the user has a clear understanding (mental model) of how these objects are used and can transfer this knowledge to the new device. Examples are spreadsheet applications like Microsoft Excel, which support typical financial activities in a way similar to a ledger sheet, but also virtual pocket calculators, or media players like the one shown in Figure 6.

Interface metaphors combine some familiar aspects of a physical entity with new behaviors and properties. We will now give an overview about common metaphors.

2.1.2.1 *Desktop*

The most prominent example is the Star user interface (Smith et al., 1982), which is based on a desktop metaphor. Information is represented through documents, which are laid out on the screen in overlapping windows similar to paper sheets on a desk. Icons symbolize office artifacts, such as folders, a trashcan, or printer. This concept has been so successful that it has become the standard for graphical user interfaces today.

2.1.2.2 *Virtual Environments*

In the entertainment domain, such as video games, virtual environments are very popular that arrange content in a three-dimensional world that can be explored by the user. The user is typically represented in this world by an avatar character, which can navigate in a walk or fly metaphor.

2.1.2.3 *Anthropomorphic Interfaces*

Anthropomorphic interfaces are mostly used for conversing and their design intentionally imitates human qualities and attributes. Humanlike animals and plants are already well known from cartoons and toys, and are becoming increasingly popular as interface metaphor in HCI. For a debate about benefits and criticism, please refer to Preece et al. (2002).

- In virtual worlds, anthropomorphic interfaces are known as **Virtual Characters**, synthetic figures that have individual personality traits. Gebhard (2009) classifies them into: i) *Avatars*, which represent human agents; ii) *VirtualHumans* that aim towards a realistic human appearance; iii) *Lifelike Characters* that create the illusion of being alive; iv) *Synthetic Actors* which appear in movies; and v) *Embodied Conversational Agents* that focus on face-to-face communication abilities, such as speech, facial expression, and body language. Figure 4 shows various virtual characters, which are employed to make systems more believable, from left to right: Microsoft's *Peedy*, IKEA's *Anna*, and DFKI's *Cyberella* and *VirtualHuman* project[4]. Peedy is a lifelike parrot from the Microsoft *Persona* project, which serves as the interface to a music-selection and recommendation system. *Anna* is a conversational agent that can be queried for product information. *Cyberella* is a virtual room

[4] www.virtual-human.org

inhabitant (Kruppa et al., 2005) that provides assistance in a shopping scenario. The DFKI's *VirtualHuman* project (funded by BMBF) includes a soccer quiz show, where human candidates play against virtual human characters.

- In the real world, **animatronic interfaces** use robotic elements for their anthropomorphous designs in order to express their emotions. They often include sensors and actuators to support additional modalities like touch and motion. Examples are Sony's robotic dog *Aibo*, Jetta's baby dinosaur *Pleo*, and MIT's robotic bear *Huggable* (see Figure 5). The bear (Stiehl et al., 2009) features a pressure-sensitive skin as input modality. It has been designed as an interactive, therapeutic companion for children and elderly in healthcare scenarios, where real animals are difficult to handle. It also provides behavioral data about patients to the nursing staff.

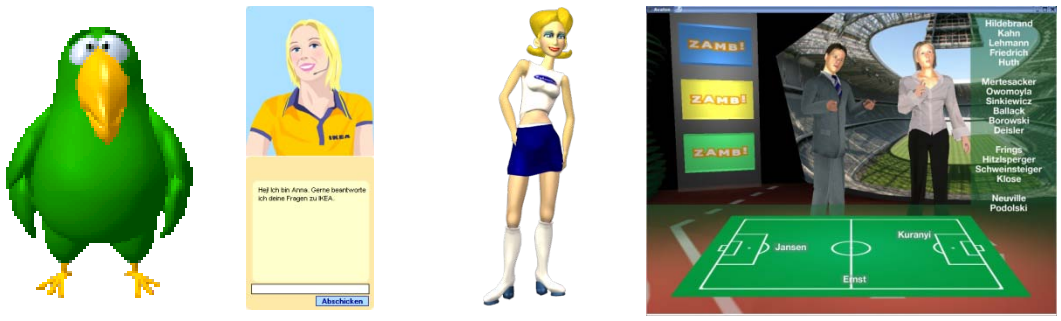

Hej! Ich bin Anna. Gerne beantworte ich deine Fragen zu IKEA.

Figure 4: Conversing with virtual characters. Images: Microsoft, IKEA, DFKI.

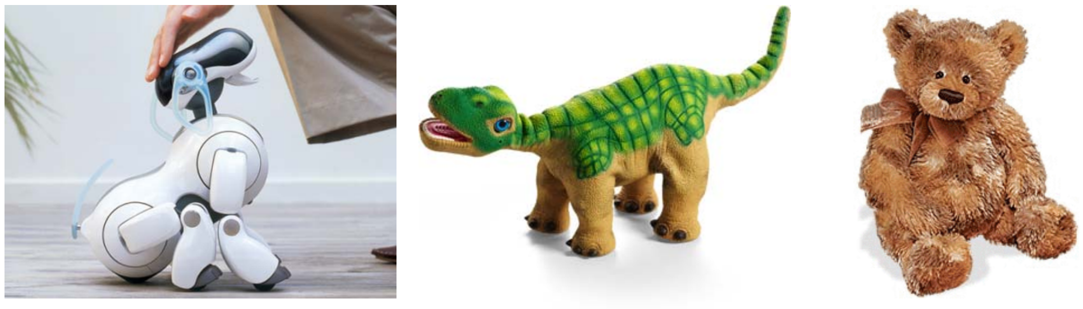


Figure 5: Animatronic interfaces: Aibo, Pleo, and Huggable. Images: Sony, Jetta Industries, MIT.

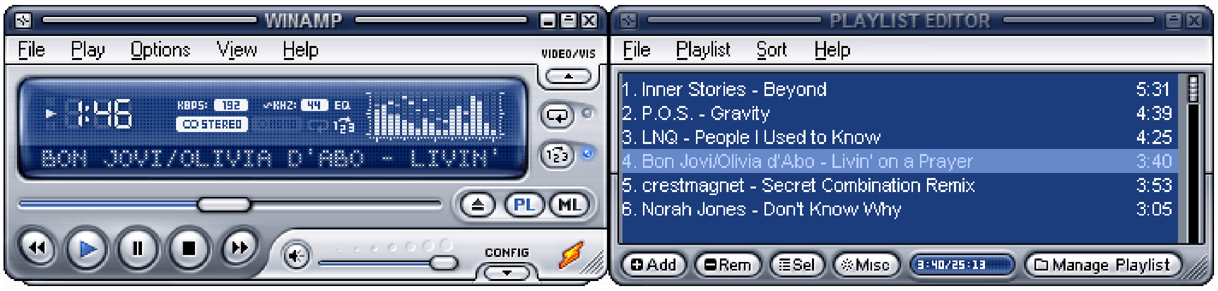
2.1.2.4 *Augmented Artifacts*

Interaction in the real world can also be implemented though computationally augmented artifacts. For example, visitors in a supermarket can interact with the *Digital Sommelier* (Schmitz et al., 2009) information kiosk through physical interaction with the wine bottles. The bottles are augmented with computing technology, such as RFID and acceleration sensors, so that they can recognize pickup gestures and which label faces up.

2.1.2.5 *Discussion*

The downside of object- and metaphor-based approaches is that, if the analogy to the real world is taken too far, the virtual user interface imitates bad designs that result from limitations in the physical world and the user interface will not tap the full potential of the digital world. In the worst case, this leads to a cumbersome and inefficient interaction style.

For example, if one is listening to music on a real CD player, navigating to a certain track requires the user to push the forward button multiple times, and jumping to a certain passage within a track can only be accomplished by pushing-and-holding the forward button to speed-up playback until the laser pickup-unit (that reads the CD) has been moved to the desired section. Whereas on a PC system with digital media, such as mp3 files, it is technically possible to directly access any title and passage within. The user interface design of a media player should accommodate for these capabilities. The media player shown in Figure 6 imitates the look and feel of a real CD-player. However, it has been extended with a playlist window, where titles can be explored and browsed and added from a music library window through direct manipulation (drag-and-drop) and a search dialog. This mode of operation is more appropriate, since titles are not grouped on physical mediums, but in a virtual hierarchy. The user interface also implements direct access to any passage through a horizontal slider.



1. Inner Stories - Beyond 5:31
2. P.O.S. - Gravity 4:39
3. LNQ - People I Used to Know 4:25
4. Bon Jovi/Olivia d'Abo - Livin' on a Prayer 3:40
5. crestmagnet - Secret Combination Remix 3:53
6. Norah Jones - Don't Know Why 3:05

Figure 6: The look and feel of Winamp's user interface imitates the display and controls of a typical real audio player (left window), but adds a slider and a playlist (right window) to allow the user to directly navigate with the mouse. Hence it combines the best of both (real and virtual) worlds.

2.1.3 Interaction Paradigms

In the previous section we have seen a variety of interaction concepts, based on objects and metaphors. They can be applied in a wide range of computing situations, from virtual worlds at the desktop PC to car navigation on the street. Figure 7 gives an overview of the most important interaction paradigms that we will discuss in the following paragraphs. The diagram horizontally arranges these paradigms according to their relationship to virtual (digital information) worlds and reality. To the left hand side, the desktop computing paradigm deals only with digital documents. The Ubiquitous Computing paradigm lies at the opposite side, since the computer is invisibly embedded into real-world objects. The Mixed Reality paradigms, such as Augmented Reality or tangible interfaces, mix digital content with real surroundings; hence they are located in the middle of the spectrum. From top to down paradigms are specialized, e.g. we consider organic interfaces as subset of tangible interfaces (each organic interface is also a tangible interface, but not vice versa). In the lower part of Figure 7, we depict the interface metaphors from the previous section according to their suitability for the virtual (e.g. WIMP) and real world (e.g. augmented artifacts). Anthropomorphic interfaces can be embodied in virtual worlds as well as in reality through animatronic interfaces.

It is worthwhile to mention that we have classified the paradigms according to the different realities since this thesis aims towards virtual models of real environments, in particular the emerging property of a Dual Reality. Alternatively, one could classify the paradigms according to other aspects like hardware considerations, their relation to social sciences, or the aspect of multimodality.

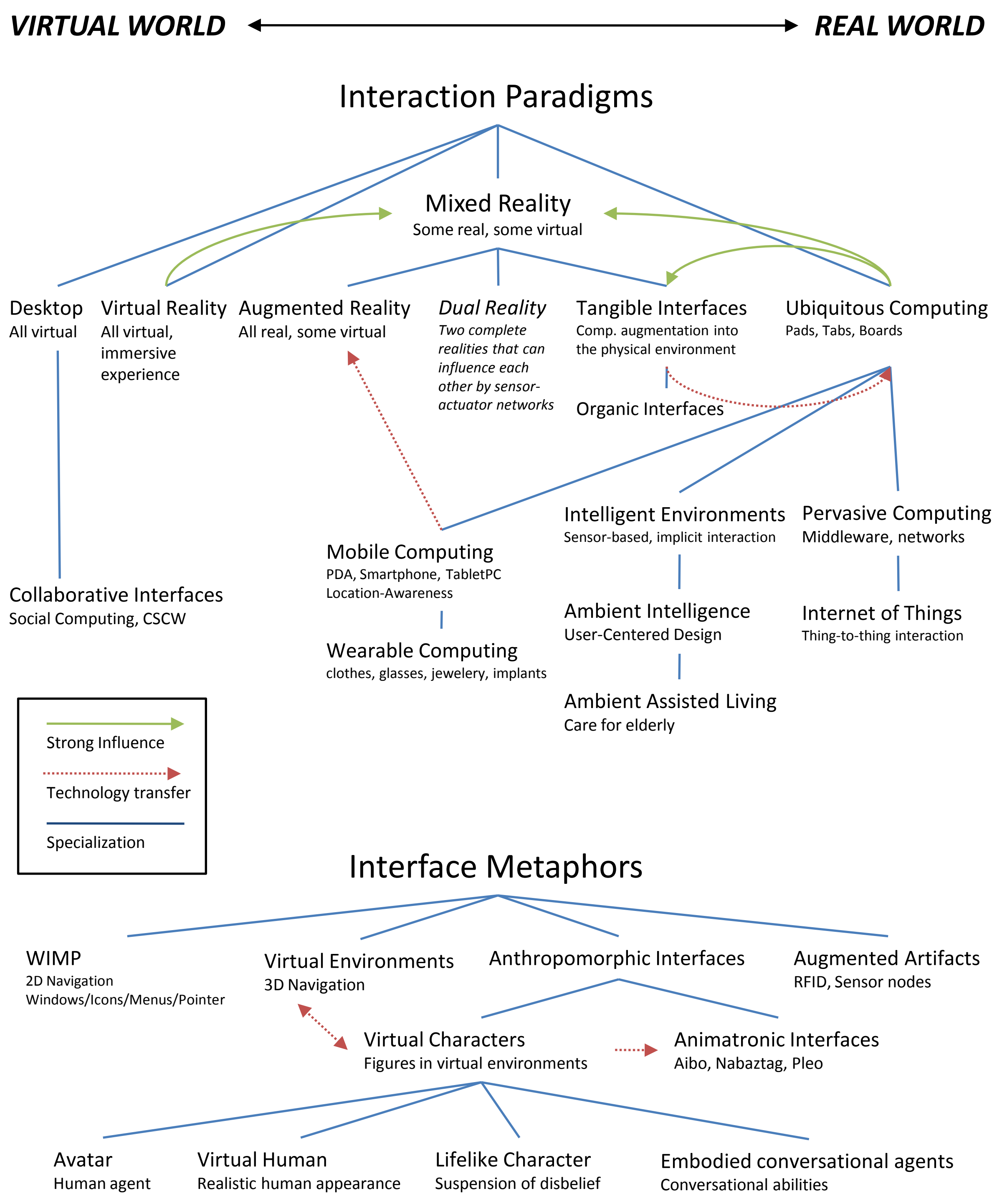


Figure 7: Interaction paradigms and interface metaphors.

2.1.3.1 *Desktop Computing*

The most common and widespread human-computer interaction (HCI) paradigm is still the desktop computer environment. Software applications are designed to be used by a single user who sits in front of a desk and interacts with the computer through a keyboard, mouse, and graphics display. The user interface of desktop applications is mostly based on the *WIMP* style (windows, icons, menus and pointers), which has been introduced by the Star system developed at Xerox PARC (Smith et al., 1982) and follows a desktop metaphor.

- **Collaborative Interfaces** aim to build systems around the familiar desktop metaphor that are accessible for multiple concurrent users and foster collaboration. In particular, the Computer-Supported Cooperative Work (CSCW) community is focused on situations of the workaday world. Collaborative interfaces are strongly influenced by social sciences and apply ubiquitous computing technologies, i.e. tangible user interfaces like multi-touch sensors and table-top displays.
- The **Activity-Based Computing** (ABC) paradigm has been introduced by Christensen and Bardram (2002) as an alternative to the desktop metaphor, which is application- and document centered. ABC is motivated by Computer-Supported Cooperative Work (CSCW) scenarios that are characterized by multiple users, mobility, and frequent interruptions, such as the healthcare domain. Bardram's thesis is that in pervasive computing environments, which are intended for such scenarios,
  "[..] the computing system must support handling human work activities directly; similar to how document-centered systems support handling documents directly. By 'work activities' we mean (more or less) well-defined tasks or processes that a person has to carry out as part of his/her job, often using computers as part of the activity." (Christensen and Bardram, 2002:107).

2.1.3.2 *Virtual Reality*

Virtual Reality (VR) aims to immersive user interfaces which let the user experience computer-simulated environments as if they were real. The virtual environment is rendered in 3-D and visualized either using head-mounted displays (HDM) that allow the user to observe the world from any viewpoint, or caves, which render an omnidirectional field of view around the user. Some VR environments also provide haptic feedback, e.g. through gloves. In contrast to Mixed Reality systems, everything that the user perceives is purely virtual.

2.1.3.3 *Mixed Reality*

In Mixed Reality (MR) interfaces some elements are part of the real world and some elements are virtual. In HCI research, three specific types of MR have gained importance: Augmented Reality, Dual Reality, and Tangible User Interfaces. Figure 8 shows three examples of driving a car in Mixed Realities; the screenshot to the left shows a pure virtual environment where the user controls a simulated car on a desktop computer. The picture in the middle shows a driving simulator that combines a virtual world display with real-world car controls for a more engaging user experience. The picture to the right shows a real traffic scene, where the car's dashboard is augmented with navigational aid.

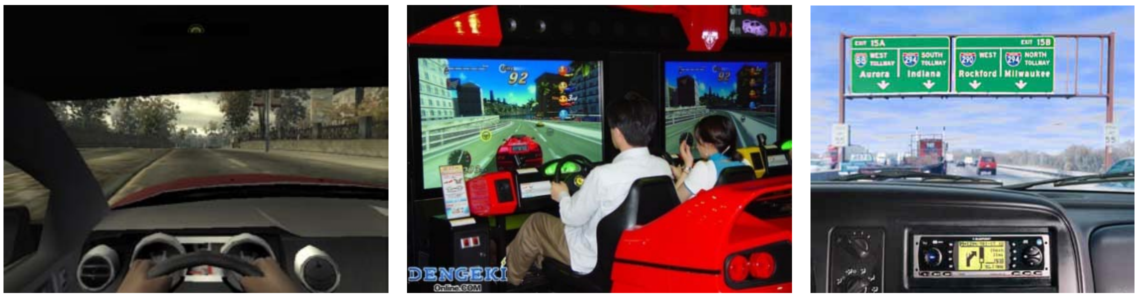


Figure 8: Driving a car in Virtual Reality (left), Mixed Reality (middle), and the real world (right).

2.1.3.3.1 Augmented Reality

Augmented Reality (AR) systems mix a view of the real world with virtual elements, usually 3-D computer graphics that overlay parts of the real world. AR systems mostly use a combination of eyeglass- or head-up display and headtracking techniques, or optical (fiducial) marker recognition. Our GPS-based pedestrian navigation system *ARREAL* (Baus, 2003) uses a small clip-on display for eyeglasses that superimposes a map of the area into the user's field of view. Other prominent applications are the *Remembrance Agent* (Starner et al., 1997) or the *MARS* navigation system (Höllerer, 1999).

2.1.3.3.2 Dual Reality

Dual Reality (DR) has been defined by Lifton (2007) as follows:

> *"An environment resulting from the interplay between the real world and the virtual world, as mediated by networks of sensors and actuators. While both worlds are complete unto themselves, they are also enriched by the ability to mutually reflect, influence, and merge into each other by means of sensor/actuator networks deeply embedded in everyday environments." (Lifton, 2007:16)*

Lifton's work aims to the automatic creation of content for online 3-D virtual worlds, such as SecondLife (see also 3.4.5), from real-world sensor streams. According to Lifton (2007) virtual worlds, as social spaces, provide an incentive for people to shape sensor data from their real lives into a form of self-expression in the virtual world. Lifton uses sensor nodes that measure light, temperature, motion, sound, and electrical current at various places in the real environment of the Dual Reality Lab[5]. The building structure of the lab has a virtual representation in SecondLife, where the sensor readings are embodied as dynamic geometric figures, such as ponds and fires, at the respective locations of the sensors.

Lifton's definition also subsumes different applications and scenarios, e.g. the driving situation that is depicted in Figure 8 (right) could also be considered as a Dual Reality, since a virtual road model is coupled to the real position of the car via a GPS sensor and provides feedback (graphical and spoken driving directions) to the user, controlling the real car.

2.1.3.3.3 Tangible User Interfaces

Tangible User Interfaces (TUIs) give physical forms to digital information and computation

[5] http://www.media.mit.edu/resenv/dual_reality_lab/

(Ishii, 2008a). The physical forms serve as both representations and controls for their digital counterparts, facilitating the direct manipulation of bits with our hands. TUIs are built upon people's sophisticated skills for sensing and manipulating their physical environments, which are not employed in graphical user interfaces. Ishii (2008b) describes typical TUI applications that combine haptic input and projected visual output, allowing the user to sculpt and manipulate digital information through tangible media, such as clay, sand, or building models, for example.

- **Organic Interfaces**. Holman and Vertegaal (2008) argue that the planar rigidity of LCD screens limits the form as well as the usability of computers and their user interfaces in terms of their possible affordances. Paper for example can be folded or rolled. The authors point out that new display technology, such as OLED or electronic ink, will allow for the construction of organic-shaped and flexible computing artifacts, which they call *blobject computers*. They define Organic User Interfaces (OUIs) as follows:

"An Organic User Interface is a computer interface that uses a non-planar display as a primary means of output, as well as input. When flexible, OUIs have the ability to become the data on display through deformation, either via manipulation or actuation. Their fluid physics-based graphics are shaped through multi-touch and bi-manual gestures."(Holman and Vertegaal, 2008:52)

Holman and Vertegaal (2008:53) suggest three design principles for OUIs:

i) Input equals output. Whereas in traditional GUIs input and output are separated (mouse and screen), input should not be distinguishable from output in OUIs: users literally touch and deform the graphics on the display. Displays should sense their own shape as input, as well as all other forces acting upon them.

ii) Function equals form. The form of an object should allow a person to perceive what can be done with it. OUIs should physically represent the supported activities.

iii) Form follows flow. OUIs should adapt their form to follow the flow of user activities in a manifold of physical and social contexts of use. Forms should always suit the activity; if the activity changes, so should the form. Rekimoto (2008) points out that interaction with future organic interfaces will also involve a more physical experience, such as illumination, air, temperature, humidity, and energy. The interaction concept will be expanded to interaction between the physical world and computers, and might include novel modalities, such as eye gaze and blowing, for instance.

A good example for organic design is given by Komissarov's *Flashbag*[6] design concept of a USB flash drive with micro pump that inflates the memory stick's volume according to the amount of data it holds. The Flashbag allows to estimate with the naked eye how much more data can be transferred into it, as shown in Figure 9.

Another example is given by Hemmert (2008), who proposes a life-like metaphor for mobile phone design. The breathing mobile phone prototype displays its status

[6] Komissarov, D., 2006, Flashbag. http://www.plusminus.ru

through permanent tactile feedback. Servo motors create a pulse signal; a calm breath pattern indicates that everything is fine, whereas an excited motion means that the phone needs the user's attention.

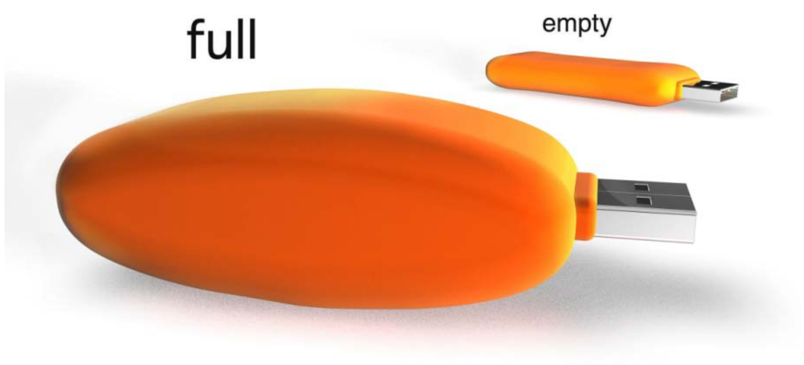


Figure 9: The Flashbag memory stick implements organic design. Image: www.plusminus.ru

2.1.3.4 *Ubiquitous Computing*

The term *Ubiquitous Computing* has been coined by Mark Weiser in his renowned 1991 Scientific America article on "The Computer for the 21st Century" (Weiser, 1991), which describes a new computing paradigm where computing devices are no longer considered to be "personal" machines, but are integrated into the natural human environment and disappear into the background. The idea of the disappearing computer is meant literally, since microcontrollers are hidden inside of appliances, but also metaphorically: we would be no longer aware of them and would use them without even thinking about them – they become "invisible" to us, like pens and scrap paper, in contrast to today's mobile devices, such as a mobile phone, PDA or notebook, which are precious items and in focus of our attention. We always carry them with us and carefully choose and customize them to express our style and personality. Weiser has predicted a future of small "scrap" computers, analogue to scrap paper, which could be grabbed and used everywhere and would have no individualized identity or importance. These small devices, to which the people at Xerox Parc referred to as "tabs", would be complemented by large, wall-mounted "boards". Finally, by seamlessly integrating computers into the real world, we would use them for our everyday tasks without consciously thinking about them.

Another important insight of Weiser has been that physical juxtaposition is basic to human perception. Thus, if ubiquitous computing devices know where they are, they can adapt their behavior in significant ways even without a hint of Artificial Intelligence. Weiser's article sparked a lot of research towards his visions, and the research community began to investigate issues such as context- and location-aware computing, sensor networks or novel display arrangements.

It should be noted that the terms Ubiquitous Computing, Pervasive Computing, Intelligent/Instrumented Environments, and Ambient Intelligence are often used as synonyms rather than different concepts and their meaning has been blurred over the years.

2.1.3.4.1 Mobile Computing

Mobile computing is concerned with computing devices that have been designed for mobility, such as personal digital assistants (PDAs), smart-phones, or Tablet PCs. Implementation of Weiser's tabs and pads. Location-awareness is an important point here, usually realized by integration of GPS receiver units, or WLAN-based positioning.

- **Wearable Computing** can be considered as a special case of mobile computing devices that are worn on the body, e.g. as necklace or wrist, or embedded in clothes.

2.1.3.4.2 Disappearing Computing

The notion of the Disappearing Computer (DC) has been introduced by the *Future and Emerging Technologies* strand of the IST programme of the European Commission[7]. DC picks up Weiser's idea of calm computing, where the computer moves into the background and becomes a secondary artifact. Streitz (2001) distinguishes between physical and mental disappearance. **Physical disappearance** means that microprocessors are integrated into artifacts so that no typical features of a computer are visible and interaction happens with the compound artifact. **Mental disappearance** happens when computers are no longer perceived as computers, but as adding interactive, communicative, and cooperative aspects to traditional everyday objects. The central question of DC is how people can interact with invisible devices. Streitz suggest a shift from information design to experience design that is focused on connectedness and awareness and realized through augmented environments. For example, the *Hello.Wall* (Streitz, 2007) implements an ambient display that connects two buildings in order to provide social awareness between the users of both sites.

2.1.3.4.3 Intelligent and Instrumented Environments

Intelligent Environments (IEs) have been originally introduced by Coen (1998a; 1998b) as highly embedded, interactive spaces with the goal to bring computation into the real world and allow computers to participate in activities that have not been previously involved with computing. IEs allow people to interact with systems in a similar way as they would do with other people. The user-interface primitives radically differ from the desktop computing concepts and replace windows, mouse and menus by gesture, speech, affect and context. Intelligent environments use cameras, microphones and a wide range of other sensing technologies to build computer vision and speech recognition/understanding systems that allow for natural, multimodal forms of human-environment interaction (HEI).

Coen distinguishes his concept of intelligent environments from ubiquitous computing by not using computationally augmented objects and artifacts everywhere. He advocates an uncumbered style of interaction without requiring that people attach gadgets like infrared badges to their bodies. Consequently, he focuses his work on computer vision.

Whereas Instrumented Environments, such as our *SUPIE* environment (Stahl, 2005), do not rely solely on computer vision, but use Ubiquitous Computing technologies to gather additional context directly from instrumented objects.

[7] www.disappearing-computer.net

2.1.3.4.4 Ambient Intelligence

The concept of Ambient Intelligence (AmI), as being developed in the ISTAG reports (2001, 2003), refers to a vision of the future, in which people are empowered by an electronic environment that is aware of their presence. People are surrounded by intelligent intuitive interfaces that are embedded in all kinds of objects and an environment that is capable of recognising and responding to the presence of different individuals in a seamless and unobtrusive way. The equipment used in such an environment is invisible or almost invisible and integrated into the background of the users' surroundings. The emphasis is on user friendliness, efficient and distributed services, user-empowerment, and support for human interactions. Examples have been developed by Philips and are described in (Aarts and Marzano, 2003; Aarts and Encarnação, 2006).

The key characteristics of Ambient Intelligence are:

- **Embedded** devices are integrated into the environment's network
- **Situated** devices can recognize situational context
- **Personalized** devices can be tailored towards the user's needs and affects
- **Adaptive** devices can change in response to the user and their tasks
- **Pro-active** systems try to anticipate the user's plans and intentions.

AmI aims at improving the quality of life by creating the desired atmosphere and functionality via intelligent, personalized interconnected systems and services. The transition to such a world requires a paradigm shift in user-system interaction. Speech and gesture recognition modalities address user benefits such as freedom of movement, and are natural candidates for such interactions. However, implementing these modalities in e.g. a home environment is radically different from implementing known methods such as buttons, dials and remote controls. Ambient Intelligence assumes a shift from desktop computers to a multiplicity of computing devices in our everyday lives, whereby computing moves to the background and ambient intelligent interfaces to the foreground.

- **Ambient Assisted Living** investigates how Ambient Intelligence technology can be applied to support the independent living of elderly persons in their home environment. Research questions comprise the recognition of Activities of Daily Living, the detection of emergency situations, and the support of care takers in their routine tasks.

2.1.3.4.5 Pervasive Computing

The paradigm of Pervasive Computing is very much based on the ideas of Ubiquitous Computing, but according to Preece (2002) the focus is more on the seamless integration of technologies and information appliances. Therefore, middleware and networking infrastructures are central research topics. Examples of networked computing devices are mobile phones such as the Nokia N series, which allow direct uploading and sharing of photos with communities through Web 2.0 (O'Reily, 2005) services. On a smaller scale, particle computers or so-called *smart dust* units combine sensors (such as temperature, acceleration, noise etc.) with low-power CPUs and basic networking capabilities so that context information about the environment can be collected and provided to applications on a higher level.

- The idea behind the **Internet of Things** (IoT) is to connect not only computers, but all kinds of everyday objects to the Web, from books to cars, from electrical appliances to food. The IoT can either be realized through the tagging of otherwise dumb objects with unique identification numbers in form of 2-D barcodes, RFID, or Near Field Communication (NFC) transponder chips, or by directly connecting smart objects to the Web via wireless LAN or 3G networks. In the former case, static information about an object can be retrieved from the Web via its ID. More intelligent features can be realized by embedding sensors into objects that obtain and log information about their environment. This approach is currently investigated by the collaborative project SemProM[8], which is part of the IKT-2020 research program of the German Federal Ministry of Education and Research the Innovation Alliance "Digital Product Memory" (DPM). SemProM develops products that use integrated sensors to keep a diary in order to make relations in the production process transparent and supply chains as well as environmental influences retraceable. The producer gets supported and the consumer better informed about the product. Furthermore, the new Internet Protocol version 6 (IPv6)[9] provides enough IP addresses to connect everyday objects and appliances directly to the Internet. This will enable sensor networks and **thing-to-thing communications**. The first international conference on the Internet of Things has recently taken place in Zurich 2008 (Floerkemeier et al., 2008).

2.1.3.5 *Discussion*

We have presented manifold conceptual models and interaction paradigms and underlined their importance for the user's understanding of how a system is supposed to be operated. It is the task of the designer to carefully choose those concepts for the user interface that best convey the designer's mental model of the system to the user.

As we have seen in the cited examples for interaction paradigms, user interface design concerns hardware and software components. Hence the generic term **Useware**[10] was introduced in 1998 by the GfA (Gesellschaft für Arbeitswissenschaft), GI (Gesellschaft für Informatik), VDE-ITG (The Information Technology Society in VDE) and VDI/VDE GMA (The Society for Measurement and Automatic Control in the VDI/VDE) to denote all hard- and software components of a technical system which serve its interactive use.

It is typical for modern artifacts to combine many of the presented paradigms in novel and creative ways. We close this Section with an interesting example, the *Nabaztag*, produced by Violet[11] (Figure 10). The *Nabaztag* is an anthropomorphous appliance in the shape of a (intentionally not lifelike) rabbit that is connected to the Web. It can act as an ambient display for real-time information, such as the weather or stock market, by moving its ears or flashing LEDs. Furthermore, it can understand spoken commands in order to retrieve information, send voice messages to friends, or playback music. Last but not least it includes an RFID reader and can be programmed by the user to perform certain actions if it is touched with everyday objects. In terms of the introduced paradigms, the *Nabaztag* is part of the Internet of Things, anthropomorphous, ambient, tangible, and recognizes augmented artifacts.

[8] http://www.semprom.org/
[9] http://tools.ietf.org/html/rfc2460
[10] http://www.useware-forum.de/
[11] http://www.violet.net

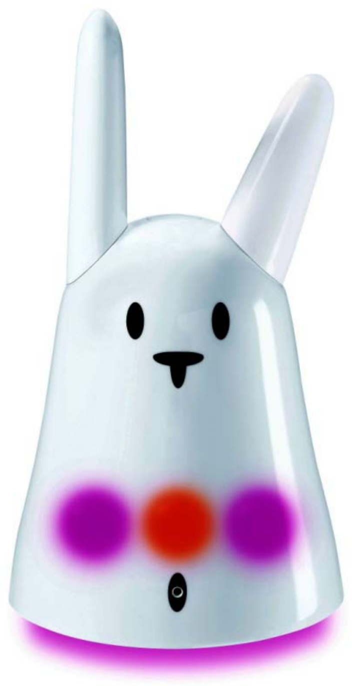


Figure 10: The *Nabaztag* rabbit is wirelessly connected to the internet and can communicate information to the user by moving its ears, flashing LEDs, reading messages, or playing music. Image: Wikimedia.

2.1.4 The Process of Interaction Design

In the previous Section we have introduced a broad spectrum of paradigms and concepts that are available today for the design of technological artifacts and instrumented environments. But which of them should be applied to a given real-world problem? This question is part of the design process, which Carroll (2000) shortly describes as follows:

> *"In design, we respond to situations in the world by creating novel artifacts to facilitate our activities and enrich our experience. Not every design effort succeeds in facilitating and enriching human existence, but each alters the world to some extent. The so-altered world presents us with new situations to which we eventually respond with new designs." (Carroll, 2000:19)*

Carroll (2000:22) further identifies six characteristics and difficult properties of design:

- Incomplete description of the problem to be addressed
- Lack of guidance on possible design moves
- The design goal or solution state cannot be known in advance
- Trade-offs among many interdependent elements
- Reliance on a diversity of knowledge and skills
- Wide-ranging and ongoing impacts on human activity

In order to overcome these difficulties, he suggests a **scenario-based design** approach. Instead of trying to control the complexity of design through decomposing the problems to be solved, scenario-based design seeks to exploit the complexity by learning about the structure and dynamics of the problem domain. Scenarios are story-like descriptions of how people accomplish their tasks by making use of a new system. Such descriptions support reasoning about situations of use, even before these situations are actually created. They highlight the suggested behavior of the system, what people try to do with it, and what procedures are adopted to achieve the user's goals. Scenarios include agents with goals or objectives, and describe what actors do in sequences of actions. Design decisions can be evaluated and documented in terms of their consequences within particular scenarios.

Instead of the **waterfall model** (Royce, 1987) of software development, where requirements definition, design, implementation, and validation are sequential stages, Carroll suggests an **iterative process** where the requirements analysis, design, and evaluation influence each other as new requirements emerge during the development process. Preece et al. (2002) point out that there usually is no single best solution to a design, but a variety of alternative designs. Therefore, the process of interaction design requires developing and evaluating alternative versions. They suggest an iterative development process that is oriented on Boehm's (1988) **spiral model** and involves four basic activities:

1) Identifying needs and establishing requirements; specific usability and user experience goals should be identified and agreed upon;
2) Developing alternative designs that meet those requirements;
3) Building interactive versions of the designs so that they can be communicated and assessed;
4) Evaluating what is being built throughout the process.

The first step is to understand what people currently do, how they act and interact with each other, which technologies they use, and what the respective strengths and weaknesses of their tools are. Different users have different needs, and interactive products must be designed to match the needs of different kinds of users. Part of the process of understanding the users' needs is to be clear about the primary objective that a new product should meet. The two top-level concerns are usability goals and user experience goals; they differ in terms of how they can be met and through which means. Usability is generally regarded as ensuring that interactive products are easy to learn, effective to use and enjoyable from the user's perspective. According to Preece (2002), usability goals can be broken down in six classes as shown in the inner circle of Figure 11. User experience goals are concerned with how emotionally attached the users are to the users. Their relationship to the usability goals is shown in the outer ring of Figure 11. Often usability and experience represent a tradeoff. For example, computer games are complex and difficult to learn, but that makes them en-joyable and fun to play. Once the user's needs are identified, requirements can be defined.

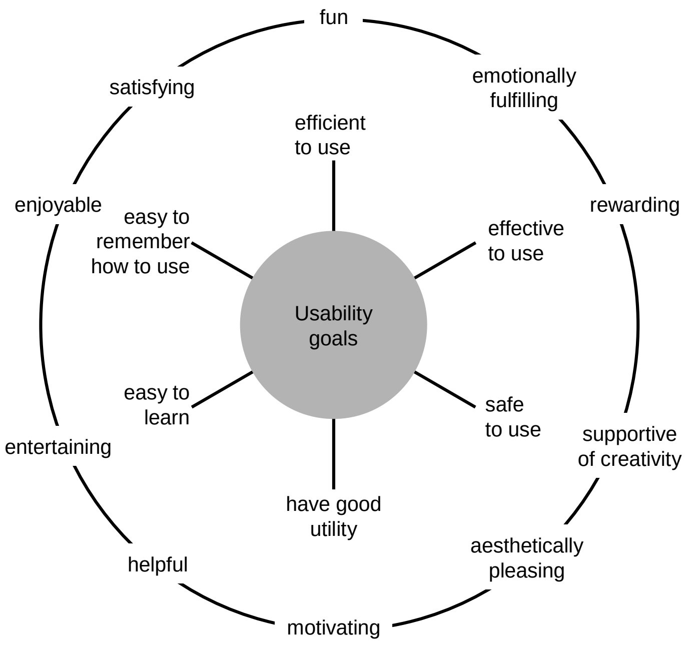


Figure 11: Usability and user-experience goals (outer circle). Source: Preece et al. (2002)

In the second step of the development process, designs have to be made that meet the identified requirements and usability/experience goals. In order to assess these designs, prototypes need to be built in the third step so that the ideas of the designer can be communicated to the actual target users in an early stage of the development process. The fourth and most important step is to evaluate the prototypes to ensure that the product is usable by the actual user. This includes various approaches such as observing users, interviewing them with questionnaires, testing and modeling their performance, and eventually involving them into the design. The evaluation phase will most likely uncover wrong assumptions about the user's needs, and the initial goals and requirements will have to be refined. Based on the modified requirements, a second prototype will be designed and presented to the user group. The development cycle repeats, until the requirements are stable and the final product fulfills the usability and user experience goals.

2.2 Activity Theory

Computer usage in the home and office today follows the desktop metaphor and is task oriented. The software application, such as a web browser, word processor or spreadsheet, provides a set of task-related tools for a single activity, from which the user can choose to perform the necessary actions towards a certain goal. Likewise, computer systems in the car are specifically designed to assist the driver's task to navigate to a certain destination place. While driving the driver might be confronted with typical situations, like decision points or slippery road conditions. These situations are well understood, and the on-board systems of a car have evolved over time to assist the driver by providing navigational aid, entertainment, and electronic stability programs for safety.

The ongoing miniaturization of technology and novel interaction devices have lead to the idea of mobile and ubiquitous computing: assist people in their everyday life environments.

The possible opportunities of technology are subject of ongoing research, and by moving away from traditional domains of human-computer interaction the possible range of activities, situations and goals vastly increases. Our experience from past research projects in the domain of ubiquitous computing indicates that it is often nontrivial and challenging to come up with useful solutions that will be accepted by a broad range of users. It is our belief that this is, among other reasons, mainly because most other situations of human-computer interaction are not as well understood as the office or driving tasks mentioned above.

Our approach in this thesis is to look at everyday life situations on a higher level of abstraction, namely the activities and needs of the user. We intend to apply Activity Theory in the sense of a wide-angle lens in order to get a better picture of what is going on in certain situations. Activity Theory provides a very precisely defined set of concepts that allows to model activity under the assumption that all activities are motivated by human needs. Activities themselves can be decomposed in to actions and goals, which in turn consist of operations. A good introduction into the concepts of activity theory as well as a systematic presentation of its principles, history and application to interaction design is given by Kaptelinin and Nardi in their book *Acting With Technology* (2006). The following sections briefly summarize chapter 3 *"Activity Theory in a Nutshell"* and chapter 2 *"Do we need Theory in Interaction Design?"* before we discuss how the concepts of Activity Theory could be applied to the design of user assistance systems. Later in this thesis (Section 6.1.1) we will pick up Activity Theory again and present a method that allows a designer to formalize and model activities in order to plan the instrumentation of everyday environments with human-machine interaction technology for user assistance systems.

2.2.1 The Main Ideas of Activity Theory

Activity Theory has been introduced through the Russian psychologist Aleksey Leontiev (1904-1979). He was a student of the famous psychologist Lev Vygotsky (1896-1934), who investigated the relationship between the human mind, on the one hand, and culture and society, on the other. Leontiev pursued an ambitious research agenda with the objective to provide a historical account of the human mind, from basic forms of psyche to advanced forms of human consciousness. In order to build a theory of the evolutionary development of the mind, Leontiev chose the concept of "activity" as an analytical tool and elaborated it during the course of his research. In his book *Activity, Consciousness and Personality* (Leontiev, 1978) he summarized the ideas and principles into a framework comprising the foundations of activity theory.

2.2.2 Basic Principles of Activity Theory

Leontiev's basic notion of activity is based on a triple made of *subject*, *activity*, and *object*. In contrast to Object-Oriented Analysis (OOA) thinking, where objects are the top-level entities and relations are identified between them in successive steps, activity is here proposed as the basic unit of analysis, thus providing a way to understand both subjects and objects. Activity of any subject is understood as a purposeful interaction of the subject with the world, a process in which mutual transformations between the poles of "subject-object" are accomplished (Leontiev 1978). All human activities are directed towards some objects; also their dreams, emotions, and feelings are directed to something. Objects can be physical things or ideal objects (in the sense of objectives), like finishing a PhD thesis, and the final outcome of an activity can change over time by the influence of external factors as the activity unfolds in a social context. According to Gay and Hembrooke (Gay and Hembrooke 2004) psychological and social objects can be ranked at the same level of importance as physical objects. This principle of **object-orientedness** implies that activities cannot exist without their objects: “any activity of an organism is directed at a certain object; an ‘objectless’ activity is impossible” (Leontiev 1981). Since the theory is historically rooted in the investigation of the development of the mind, the subject always refers to a living thing, which has either basic (animal or human) *biological needs*, such as the need for food to survive, or more abstract human *psychological needs*. The object might either be something that directly satisfies the need, or some tool that indirectly helps to acquire the desired object, such as an axe or computing device.

Thereof results the principle of **hierarchical structure of activity**. An object that directly meets a certain need has the status of a *motive* that motivates the activity of the subject, whereas all secondary objects are considered as *goals* towards this motivating object. The steps which are necessary to achieve these goals (that are different from the motivating object) are called *actions*, which in turn can be refined on a lower level through *operations*. They identify automated routine processes which establish certain *conditions* that are required to execute an action, and people are usually not aware of them. The hierarchical relationship between activity, actions and operations is depicted in Figure 12. A single activity can consist of multiple actions, and each action can be made of multiple operations.

Activity	Motive
Actions	Goals or Aims
Operations	Conditions

Figure 12: The structure of human activity, according to Activity Theory.

The distinction between the subject’s motive and goals reflects the division of labor which is one of the key aspects of human culture. Leontiev introduces the concepts of motive and goal by the example of a primeval collective hunting scenario:

> *"There is no activity in animals that does not [[correspond]] to some sort of direct biological need [[....]] Let us now examine the fundamental structure of [[human]] activity in the conditions of a collective labour process [[...]]. When a member of a group performs his labour activity he also does it to satisfy one of his needs. A beater, for example, taking part in a primeval collective hunt, was stimulated by a need for food or, perhaps, a need for clothing, which the skin of the dead animal would meet for him. At what, however, was his activity itself directly aimed? It may have been directed, for example, at [[the goal of]] frightening a herd of animals and sending them toward other hunters, hiding in ambush. That properly speaking, is what should be the result of the activity of this man. And the activity of this individual member of the hunt ends with that. The rest is completed by the other members. This result, i.e. the frightening of game, etc. understandably does not in itself, and may not, lead to satisfaction of the beater's need for food, or the skin of the animal. What the processes of his activity [a.k.a. his actions] were directed to did not, consequently, coincide with the motive of his of his activity; the two were divided from one another in this*

instance. Processes, the object and motive of which do not coincide with one another, we shall call 'actions'. We can say, for example, that the beater's activity is the hunt, and the frightening of game his action" (Leontiev, 1981: 209-210).

In Activity Theory, all levels can move up and down, as conditions change and the subject's abilities to perform actions develops unconsciously (which separates actions from operations) over time through learning and forgetting. Actions may eventually become operations as they do no longer require consciousness and concentration. This interrelation is covered by the principle of **internalization-externalization**, which differentiates between internal and external activities. Internalization is the transformation of external activities to internal activities, as external components can be omitted through learning and mental images. For example, through practice a typist builds a mental model of a keyboard and does not have to look for the keys in order to enter text. Externalization describes the opposite effect and is necessary when an internalized action needs to be scaled; for example when many telephone numbers need to be recalled we have to write them down. External tools are also required to coordinate collaboration between several people, for example a to-do list or calendar. Externalization is also necessary if an operation fails. In case of such a "breakdown" situation, operations like entering a password unexpectedly fail due to exceptional conditions or changes in the environment, for example when the keyboard layout is changed. The operation then suddenly becomes an action or even a new activity.

The principle of **mediation** refers to the special role of culturally developed artifacts in human activities, such as tools or signs. The shape and material of a tool, together with knowledge how to use it, reflects the accumulated experience of many other people who have invented or improved the tool, sometimes over generations. Gay and Hembrooke (2004:6) point out that tools and artifacts can be transposed into object status and vice versa. This might happen if the tool suddenly breaks as the subject is using it. In that case, the tool becomes the object of a repairing action.

Further principles include the *interpsychological and intrapsychological* stages of the development of mental abilities, but these are outside the scope of the thesis. Additionally, human interaction should always be analyzed in the context of *development*.

2.2.3 Engeström's Activity System Model

Since the basic framework of activity theory has been published by Leontiev in 1981 it has been extended in various ways. Although we do not apply it in out thesis, it is worthwhile to mention Engeström's (1987) **activity-system model** that has been adopted by many as post-cognitive theory for HCI research, as we will see in the following sections. Engeström argues that individuals can carry out actions only within a larger-scale activity system. Therefore, his activity-system model extends the subject-object relationship of Leontiev's theory from individual activities to a collaborative phenomenon by introducing the concept of *community*. The resulting model describes interactions as triples between subject, object and community. Each of these interactions can be mediated through instruments, rules and division of labor. The resulting model is typically represented as a triangle as depicted in Figure 13.

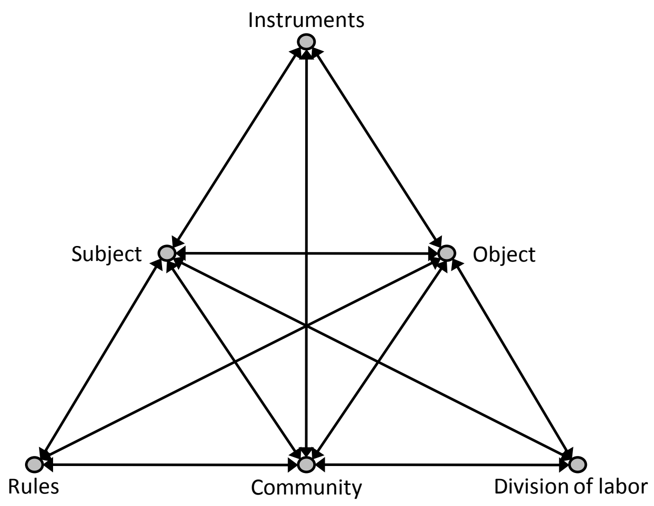


Figure 13: Engeström's activity system model introduces the concept of community.

2.2.4 Activity Theory in Interaction Design

Kaptelinin and Nardi (2006) argue that the two traditional approaches to human-computer interaction, the cognitivist model and ethnomethodology, both have their weaknesses as the focus in design is shifting from a single-user desktop environment to collaborative uses of technology with multiple actors and objects, varied virtual and physical contexts, an expanded set of activities, and human experience in general. Examples are distributed work scenarios, multiplayer online games, and education. Even though Kaptelinin does not explicitly mention the Ubiquitous Computing paradigm, his argumentation clearly subsumes it.

The cognitivist approach adopts the information-processing paradigm of computer science as the model for human cognition, and can be easily converted to design, but is too narrow in its scope. It considers only interaction between the user and the system on a task-level, and does not consider higher-level goals and motives or social aspects. By contrast, ethnomethodological accounts provide rich depictions of practice, which emphasize social relations and activities besides human-machine interaction, but they are difficult to relate to the designers' concerns.

According to Kaptelinin and Nardi, Activity Theory could fill the gap between both approaches since it is rich enough to capture the most important aspects of the actual use of technology and descriptive and generalizable enough to be a practical, useful tool in interaction design. They suggest Activity Theory as a *"post-cognitivist perspective [on interaction design] that incorporates the complexity of real practice, widening analysis to include a cycle of evaluation and design in which people and artifacts influence one another"* (Kapelinin and Nardi, 2006:25). Such a theory will be helpful for the design of systems that address: i) collaborative use by groups and the larger society, ii) varied physical and virtual contexts, iii) an expanded set of activities, and iv) human experience in general, not just cognition.

Activity Theory extends the scope from "user-system" interaction on a task-level to more general "subject-object" relationships in a larger context of human activity, and considers social aspects besides the pure functionality of a system. It also takes into account long-term developmental transformations in users and artifacts, so that the cycle of evaluation and design can be analyzed. In summary, activity theory could be of enormous help in the design of complex systems with multiple actors and objects, since it identifies fundamentally important concepts, explains relationships and processes, and facilitates creativity, invention, and discovery.

2.2.5 Applying Activity Theory to the Design of User Assistance

The concepts of activity, actions and operations can be directly applied by the designer of a user assistance system in order to hierarchically analyze and model the users' potential needs for assistance on multiple levels of abstraction. The designer can anticipate the required functionality of an assistance system with regard to the users' needs, motives and objects of desire, on the one hand, and the users' need for assistance in using the system itself, on the other hand.

On the top-level the subject (the user) is motivated by biological or psychological needs and desires certain objects. In a real-world scenario, typical needs could be eating, resting, excitement, or social experience. Corresponding top-level objects of desire could include food, a seat, a thrilling movie, or a friend. It is typical for everyday life settings that the user interrupts an ongoing activity in favor of immediate needs, e.g. to buy a drink or take a photo. The interleaved performance of activity often results in forgetting of details and dates, hence people tend to externalize such information. Personal information management (PIM) devices assist to plan and choose between multiple activities, and provide reminders. Activity Theory could be applied to model the hierarchic structure of activities. If the desired top-level object is for some reason not immediately available, the subject has to plan and take the necessary actions, such as withdrawal of cash from an ATM, or ticket reservation for a movie. Each of these steps in turn could include a navigational sub-goal, which recursively requires search or exploration actions; the user is likely to consult navigational assistance. On the lowest level of activity, the user performs internalized operations. For example, typing in the PIN number on the keypad of the ATM becomes often a gesture that is learned trough frequent use. It does not require sequentially recalling and looking up each of the four digits. Using a cell-phone to call someone from the contact list can be considered as an operation as well, as it takes a fixed sequence of keys. However, in case that something goes wrong, for example the dialogues appear in a foreign language, the full attention of the user is required in order to recover from the breakdown, and the operation transforms into an action. In that case, the user requires assistance on the lowest level, e.g. how to adapt the language of the user interface, in order to proceed.

2.2.6 Activity-Centered Design

Norman (2005) suggests an Activity-Centered Design approach as an alternative to User-Centered Design, where the designers of an artifact use their own understanding of the activities to be performed to determine how the device would be operated. Norman refers to his version of the hierarchical structure of activity and points out that to him an activity is a coordinated, integrated set of tasks. Norman's argumentation directly relates to Activity Theory's principles of tool mediation and development: the shape of a tool reflects cultural experience and knowledge about an activity, and by learning how to use a tool, the user learns the activity itself.

Gay and Hembrooke (2004) use Activity Theory as an orienting framework for context-based design, called Activity-Centered Design (ACD). The current development towards pervasive and ubiquitous computing applications leads to a shift in HCI design from a focus on human-computer interaction to a focus on human interaction that is mediated by technology in an everyday context. Accordingly, Gay and Hembrooke anticipate a shift from user-centered design towards context-based design. Their goal is to develop models and methods that support empirical investigations of HCI in Ubiquitous Computing environments and allow applying these findings to the design and evaluation of new technologies. Particularly the portability of wireless computing devices allows for flexibility and accessibility across temporal and spatial boundaries. The authors argue that requirements analysis and system evaluation should take place in real settings and consider the social context of activities, since most tasks involve cooperation and communication with others. The designer of a system should also understand the changes that are caused in an organization or environment by the introduction of new technologies.

Gay and Hembrooke follow Engeström's activity system model as a theoretical foundation for their work, to which they add an ecological perspective that looks at a system in the broader context of interacting systems. Component systems within ecological systems are characterized by progressive, mutual accommodation and extinction throughout the life of the system. Ecological systems are not always harmonious and functioning but have constant tensions, discontinuities, and breakdowns that are necessary for survival and adaptability. The tensions and breakdowns can be used as points of reference for understanding and describing design. The authors suggest an iterative design cycle that reflects the ecological perspective on system design, as shown in Figure 14. First, researchers and designers examine current activities and practices. Next, they identify tensions and conflicts within activity systems and develop new solutions and designs. The resulting system is evaluated in actual settings, and the development cycle repeats with a revision of the requirements and a redesign of the system.

The authors use the *SCOT* (Social Construction of Technology) framework as concrete implementation of the interactive design process and describe in a case study how it has been applied to the design of a museum guide. In the first phase, an initial needs assessment has been conducted to determine the objectives of the guide to be developed, such as navigation, messaging, and artist information. The SCOT approach helped to consider the different social perspectives of multiple groups such as the project's funders, museum staff, administrators, designers, and patrons. It helped to identify and consider conflicts of interest and to implement and evaluate prototypes that match the interests of all groups. In a second iteration of the cycle, the needs have been assessed based on the earlier experience with the prototypes, and led to a different ranking of objectives.

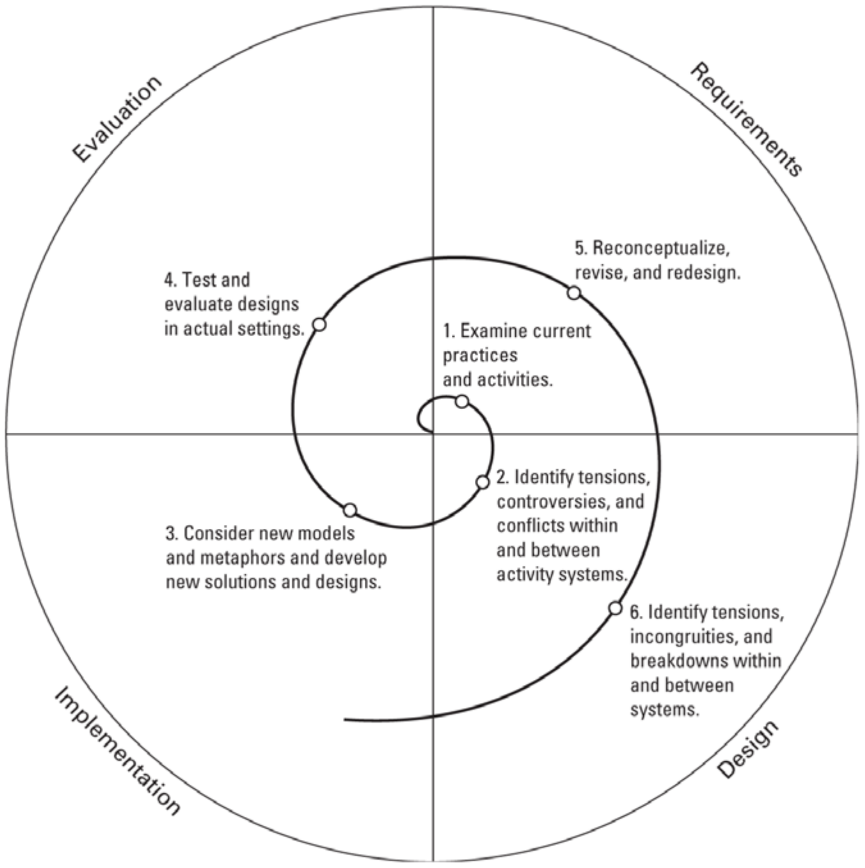

Evaluation
Requirements
Implementation
Design
1. Examine current practices and activities.
2. Identify tensions, controversies, and conflicts within and between activity systems.
3. Consider new models and metaphors and develop new solutions and designs.
4. Test and evaluate designs in actual settings.
5. Reconceptualize, revise, and redesign.
6. Identify tensions, incongruities, and breakdowns within and between systems.

Figure 14: An iterative design cycle that anticipates the cyclic process of change.
Source: (Gay and Hembrooke, 2004).

2.2.7 Taskonomy vs. Taxonomy

In one of his essays Norman (2006) compares two opposing approaches on how the menus of mobile phones can be organized, namely taxonomy and taskonomy.

The traditional **taxonomy**-oriented approach clusters logically similar functions together, which are in turn hierarchically grouped. The resulting tree-like structure allows quick access to certain functions or information. This schema is easy to understand and works well for libraries, stores, or operating systems.

However, taxonomy does not take into account activities and their sequences of actions. In the context of a certain activity, it requires too much overhead to reach different leaves of the menu tree. It is more efficient if things which are used together are placed next to each another. Anthropologists Dougherty and Keller (1985) have analyzed such a task-oriented arrangement of tools in the workshop of a blacksmith and coined the term **taskonomy** for it.

Norman (2006) argues that in the case of mobile devices, such arrangements support activity patterns, like taking a picture and sending it as a multi-media message, better than taxonomic menu structures. Norman concludes that the best solution is to provide both solutions. All items should be logically and sensibly ordered in taxonomy, but once a certain item has been selected, additional taskonomic information should appear.

This principle has been implemented Microsoft Windows through context menus. For example, if the user right-clicks on a file icon in the file explorer, a context menu appears that provides a set of applicable actions, depending on the type of the file. Text documents can be converted to PDF, images can be rotated or used as desktop wallpaper, and media files can be queued for playback in the media player. Examples for context-dependent menus can also be found in some mobile phones which offer a send-to option in the context of photos.

It is notable that the same principle can also be applied to printed user manuals. Carroll has already recognized in 1990 that users don't read software documentation, since the typical menu-oriented style of writing requires them to go through the whole manual, in order to solve even simple tasks. Hence Carroll (1990) suggests a minimal, task-oriented style of writing for operating manuals that today has been well adopted by the industry for the documentation of mobile phones.

2.2.8 Activity-Based Computing

The concept of Activity-Based Computing (ABC) has been proposed by Bardram (2004) as a new way of thinking about supporting human activities through mobility and collaboration in Pervasive Computing environments. Whereas ABC has a certain similarity with Leontiev's Activity Theory in that activity is the primary concept and that activities are hierarchically structured, ABC follows its own principles and provides a framework for the implementation of applications. Bardram has developed ABC as middleware system and a programmer's interface that manages computational activities. Bardram (2004:57) defines

- **Activity** as an abstract, but comprehensive, description of the run-time state of a set of computational services. An activity is collaborative, i.e. it has a number of participants;
- **Activity-Based Computing** as a computing infrastructure, which supports suspending and resuming activities across heterogeneous execution environments and supports activity-based collaboration.

Bardram notes that there is no explicit link between the computational activity and a human intent; the link to the intent of the activity is left for the users, or their organizational setting, to establish.

The background and motivation for the development of the ABC paradigm have been studies of healthcare practices in hospitals. According to Bardram (2005), the work in hospital is fundamentally different from office work; extreme mobility, ad hoc collaboration, frequent interruptions, and a high degree of communication make healthcare an interesting application area for the design of pervasive computing technology. Clinicians are traditionally faced which a lot of different systems, and carrying out a single activity may involve the handling of many different services or applications. For example, if a doctor prescribes medicine for a patient, the doctor reviews the medical history from a database, looks over the medicine schema, studies the results of a blood test and views X-ray images, etc. From this example, Bardram identifies three levels of abstraction. On top is the *activity level* of human activities. Below is the *application level* of computational services and the lowest *data level* includes files or material being manipulated. The aim of Bardram's ABC concept is to directly support work on the activity level through multiple devices, whereas contemporary computing systems only support tasks on the application level.

According to Bardram's principles of Activity-Based Computing (2005), activities can be broken down along three dimensions: i) task and material, ii) time and space, and iii) users.

Human activity is accomplished by carrying out a range of tasks, which use or manipulate some material. According to these principles of ABC, a **computational activity** reflects a task as a collection of computational tools for carrying out tasks (applications) and data (files or documents). By doing so, computational activities become first-class entities that are represented explicitly. This means that activities are managed and persistently saved over time, and are directly accessible to users. A user can alternate between a range of activities, which can be suspended at one host and later resumed at another host by restoring the state of the activity. This concept supports interruptions and multi-tasking in work and mobility in time and space. Activities are also inherently shared between multiple participants and are thereby collaborative. The list of participants implements an access control mechanism to the activity; only listed participants can resume the activity.

2.2.9 Summary and Discussion

Motivated by our research question of how to achieve a seamless integration of multiple user assistance services in instrumented environments, we have investigated the theoretical background of human activity.

In Section 2.2 we have introduced the main concepts and principles of Activity Theory according to the Russian psychologist Leontiev (1981). Activities are motivated by human needs, and hierarchically structured in actions and operations. The theory reflects the principle of division of labor in that it differentiates between activities that are directed towards the user's needs and actions that are targeted towards secondary goals. Operations represent internalized actions that do not require conscious thought and planning. For completeness, we have also mentioned Engeströms (1987) extension of the theory around the concept of community, which is often used to represent social aspects of technology.

Regarding the application of the theory to interaction design, we have cited Kaptelinin and Nardi's motivation for a new, post-cognitive theory that is focused on multiple users and objects in everyday-life scenarios. Following their argumentation, we have summarized Gay and Hembrooke's (2004) Activity-Centered Design framework. Finally, we have introduced other activity-related concepts, such as taskonomies and Bardram's (2004) Activity-Based Computing framework (ABC). ABC has been designed to support clinicians in hospitals to manage a set of parallel and cooperative activities by grouping documents into sets of computational activities.

Whereas the studied theories shed some light on the concepts and structure of human activity, they can't immediately be applied to the design of proactive user assistance systems in instrumented environments. Activity Theory is informal and lacks a structured modeling method, and the ABC framework is targeted specifically towards computational support for collaborative work processes in multi-tasking environments.

In order to realize our goal of developing a structured method that guides the designer of user assistance systems in their decisions, we need to further formalize human activity so that it can be represented through modeling tools and applied during the development process. We have already made the first step towards this goal by giving a definition for what a *situational semantic activity model* is in the introduction of this thesis (see 1.4). We will further elaborate this definition later in Section 6.3.3 and present a set-based, tabular representation of an activity model, which can be directly implemented through modeling tools as the one that we are going to introduce in Section 7.3 of this thesis.

2.3 Wayfinding and Spatial Knowledge

We have seen in the previous section that according to Activity Theory every activity is motivated by the need for an object. This motivating object is seldom ready at hand, so the user needs to physically move to the place where the desired object can be obtained, produced, or used. The user therefore has to plan the appropriate sequence of actions to get there. If the destination is located in a different room or even building, this may be a nontrivial task for the user, particularly in unfamiliar environments with no or little spatial knowledge. Hence it is an important aspect of user assistance systems to provide some navigational aid to support the user during the planning and execution phases of their way.

In this section, we describe the basic concepts of wayfinding, a term that has been coined by Lynch (1960) and refined by Arthur and Passini (1992). From a cognitive science perspective, spatial orientation is based on the ability to form a *cognitive map* that is an overall mental image or representation of the spaces and the layout of a setting. You are considered spatially oriented if you have an adequate cognitive map of the surrounding setting and are able to situate yourself within that representation. But people without cognitive maps do not feel disoriented and are still able to find their ways. The terms *spatial orientation* and *cognitive mapping* only describe the static relationship of a person to their spatial setting, but they cannot encompass the dynamic aspects of their movement. In comparison, **wayfinding** describes the process of reaching a destination, whether in familiar or unfamiliar environment. It is best defined as spatial problem solving under uncertainty, and comprises three steps:

1) Decision making: development of a plan.
2) Decision execution: appropriate behavior at the right place in space.
3) Information processing: environmental perception and cognition

In the decision making phase, the user develops **hierarchically structured decision plans**, which involve different means of transportation on every level, from intercontinental traveling by airplanes to short walking distances. Let's assume that a user, who is living in Saarbrücken, intents to spend a week in Corfu, so the highest level of planning involves searching for affordable flights. Let's assume that no direct flight is available from the regional airport, so the user must choose to depart from either Cologne or Stuttgart. The next step would be to prepare driving directions from Saarbrücken to Cologne airport, which offers cheaper long-term parking opportunities. Finally, the user would consult a map of the airport to plan the route from the car park to the check in counter. Unfortunately, it is usually not possible to create a fully detailed plan a priori. At the time of planning, it is unclear where the check-in counter will be located and which gate will be used for boarding the plane, since these locations are assigned dynamically depending on the current situation, such as delayed flights. Therefore, the resulting plan will include decisions to look up information from signs and displays in situ. The decision part for such a route could look like Table 1. It is notable that the hierarchical nature of the decision plan is closely related to the hierarchical structure of Activity Theory, which has been introduced in the previous section of this chapter. Considering our example, the motivating object of the vacation activity could be the beach, and it requires a sequence of actions to go there: drive to the airport, park the car, find and go to the terminal, check-in and board the plane, and so on.

Table 1: Wayfinding decisions, according to Arthur and Passini (1992).

D1	to go to the check-in counter in terminal 2				
		D2	to park the car		
				D3	to park the car on a reserved spot
				D4	to leave the car park
		D5	to go to terminal 2		
				D6	to consult a map
				D7	to follow signs
		D8	to find the address of the check-in counter		
				D9	to go to the main entrance
				D10	to consult the display
				D11	to follow signs
				D12	to check-in
		D13	to go to the gate		
				D14	to consult the boarding pass
				D15	to follow signs

According to Arthur and Passini (1992), the *wayfinding decisions* of the resulting plan are composed of two parts, the behavior and environmental entity. A *behavior* is a simple action, such as turning right, going up, or looking up information from a map or timetable. An *environmental entity* is the according object (aim or goal) of the behavior, such as an intersection, stairs, or billboard.

In the execution phase, the user matches their perception with their cognitive map which situates the decisions in space (such as the mental image of the buildings in a cityscape), and executes the according behavior. In doing so, information processing takes place and depending on the type of user, two different strategies of cognitive mapping can be distinguished:

- **Route maps** represent the environment from an egocentric perspective in terms of routes. People memorize sequences of points where they change direction, the according angles and the distances between these points. These maps are usually based on personal exploration and have a schematic and abstract *propositional* representation without orientation.
- **Survey maps** represent topographical relationships between elements in a figural or *analogue* representation. This type of map is usually learned from using physical maps and tends to be oriented and precise, but less flexible than route maps.

Both types of cognitive maps rely heavily on **landmarks** as anchor points. People may begin the cognitive mapping of new environments with landmarks and fill in routes between them, or vice versa. Landmarks are things that we remember, such as concise buildings or places in a cityscape. According to Lynch (1960), landmarks are objects which, due to their uniqueness, are chosen as the most distinctive features of different areas within a city. Arthur and Passini (1992) define four factors that enhance memory of buildings as landmarks: The form of the building (size, contours, complexity of shape, and architectural style), visibility and access, use, and symbolic significance. Sorrows and Hirtle (1999) likewise define: i) a *visual landmark* is an object that is a landmark primarily because of its visual characteristics; ii) a *cognitive landmark* is one in which the meaning stands out; and iii) a *structural landmark* is one whose importance comes from its role or location in the structure of space. Lynch (1960) also distinguishes **distant landmarks** that are visible from many parts of a city, but not used in giving local directions. Landmarks can be point-like (e.g. a building), linear (e.g. a river), or areal (e.g. a historic quarter of a town). Krieg-Brückner et al. (1998) also distinguish between the uses of landmarks; Landmarks are used to determine the position of an agent in space, whereas **routemarks** are used to determine an agent's postion along a route.

2.3.1 Complexity of Wayfinding problems

Raubal (2002) suggests the use of **image schemata** in order to model the process of human wayfinding behavior from a cognitivist's perspective. According to Raubal, the complexity of a given wayfinding task depends on the choices and clues. A **decision point** is defined as a place where more than one path can be taken. Clues are signs that are visible from the decision point, and can be rated either as good (clear relationship to the wayfinding destination), poor (litte relation), or none. Problems are mainly caused at decision points with none or poor clues. Raubal also stresses that people expect clues even at non-decision points; they might stop following a path if they find no confirmation information that they actually are on the right path, and return to the last decision point.

2.3.2 Route Directions

In the above paragraphs we have described wayfinding as a process of decision making. Navigation systems ease this task by providing route directions to the user. Such directions describe a route as a sequence of instructions that convey which actions to take at decision points along the suggested way from A to B. They can use various media and modality, e.g. text, speech, graphics, and maps. Even tactile feedback has been applied to guide users.

Route directions can be divided into two classes: i) **complete** descriptions and ii) **incremental** descriptions (Maaß, 1993). The prior describe the whole route and are given to a questioner in situations where no further information is assumed to be available during the navigation task. This is for example the case if a local person describes a route for a tourist. Of course, the questioner must memorize the instructions correctly. Another typical case is that someone who does not own a mobile navigation system downloads and prints driving directions from the internet in order to prepare for a trip. In contrast, if a co-driver or navigation system is available, incremental descriptions are given dynamically as the driver approaches a decision point. Maaß (1996) has developed a cognitive model and the agent-based navigation system *MOSES* that automatically creates multimodal incremental route descriptions ("follow the road", "drive along the grey building", "take the second exit", ..).

Richter and Klippel (2005) discuss different properties and elements of route descriptions. Routes for cars e.g. can be optimized either to describe the shortest or fastest path. Alternatively, routes can be optimized to reduce the number of decision points. Regarding the reference system, instructions can be given: i) from the perspective of the wayfinder (egocentric reference); ii) with respect to elements of the environment (allocentric reference); or iii) with respect to fixed references outside the environment (absolute reference). According to the frame of reference, directions can refer to cardinal directions (e.g. 'north'/'south') or distant landmarks, structural features like slant ('uphill'/'downhill'), or local landmark objects (e.g. salient buildings) and annotations (e.g. street signs). Richter and Klippel (2005) further suggest adapting route directions to the actual path to take by chunking together decision point-action pairs, e.g. "turn right at the third intersection".

2.3.3 Macro-Navigation vs. Micro-Navigation

The process of wayfinding and navigation has been so far concerned with the problem of moving from a starting point A to a destination point B, in terms of buildings or rooms. However, wayfinding typically occurs in the context of some activity that is motivated by the need of an object. In the introduction of this thesis we have described a scenario (see 1.2) where a traveler at the airport intends to buy a digital camera. In that case, the primary navigational task is to find an electronics store inside the airport. However, once the traveler has entered the shop, the navigation continues on a smaller scale as the traveler has to locate the camera on the shelf. The latter task is inherently different from the prior, since route directions are no longer helpful, yet assistance may be desired, since it takes cognitive effort to find a specific item within a collection of similar objects. To differentiate between these two types of navigation tasks, we have introduced the concepts of macro-navigation and micro-navigation in (Stahl et al., 2005):

- **Macro-navigation** is concerned with tasks, in which the navigational goal is beyond the user's perception of the current environment, in the sense that the user has to move through the environment to reach their destination, which requires route and survey knowledge about the environment.
- By the term **micro-navigation** we conceive the task to focus the user's attention to a spot within the perimeter of their perception, e.g. an interesting product in the shelf. Macro-navigation tasks can be decomposed into a sequence of micro-navigation tasks (transitions from one range of perception to another).

It should be noted that, according to Montello's (1993) definitions of psychological spaces, macro-navigation takes place in **environmental space** that is "too large and otherwise obscured to apprehend directly without considerable locomotion" whereas micro-navigation takes places in **vista space** that can be "visually apprehended from a single place without appreciable locomotion" (Montello, 1993:315).

2.3.4 Object Localization

Particular in the context of micro-navigation, object localization plays an important role. In our instrumented environment, we use visual cues to focus the attention of the user to certain objects, like the desired camera in our scenario. If the preferred modality is text or speech, the location of objects has to be verbalized using spatial relations. Based on empirical studies, Gapp (1997) has developed the *OLS* system that is able to compute cognitively adequate descriptions of an object's location in relation to other salient reference objects. Gapp's system can be applied to macro-navigational as well as micro-navigational contexts. In a scenario of campus navigation *OLS* can answer requests like *"Where exactly is the math building located?"* with descriptions like *"The math building is located in front of the computer science department"* (Gapp, 1997:183).

2.4 Location Models for Ubiquitous Computing

How can we model and represent spatial knowledge in the computer, so that a user assistance system can perform a local search for points (or objects) of interest, derive routes, and present appropriate wayfinding instructions to the user?

Ideally, such a navigational aid system would automatically generate and combine graphical and textual elements into different presentations, which satisfy both types of users: those who prefer route maps and others who like survey maps. According to Becker and Dürr (2005), location models for ubiquitous computing applications can be classified by their coordinate systems; geometric coordinate systems represent the position of objects by tuples of real numbers, whereas symbolic coordinate systems use abstract symbols. Besides, hybrid models exist which combine numerical and symbolic coordinates. In the following sections, we will discuss the general requirements for location models and introduce the basic concepts and examples for geometric, symbolic, and hybrid models.

2.4.1 Requirements for a Location Model Space

Becker and Dürr (2005) list five applications of location models in the ubiquitous computing domain: i) position queries; ii) nearest neighbor queries; iii) navigation; iv) range queries; and v) visualization.

All cases, except navigation, can be handled well with geometric location models and Euclidean distances. Outdoors, geographical features are best modeled using spherical coordinate systems, like longitude and latitude, since GPS is used for positioning. Indoors, local Cartesian coordinate systems are more convenient to model buildings and locations according to ground plans or own measurements. Sometimes it is also practical to use pixel coordinates of a digital image, such as a ground map or aerial photograph. In order to provide navigational aid, Cartesian coordinates are also simple to visualize in maps.

Navigational tasks require additional knowledge about the connectivity between places. According to Becker and Dürr (2005), it is not sufficient to represent the geometry of roads or walls, but the system also requires qualitative information about how to get from one location to another. This semantic knowledge has additionally to be represented by topological relations like *<connected-to>*. The model should also provide additional information about connections regarding the quality of a route, e.g. speed limits for cars or its suitability for persons in a wheelchair.

It is noteworthy that for the pedestrian navigation domain special requirements to the location model arise from the pedestrians' high degree of freedom in comparison to cars. Whereas it is for car navigation applications sufficient to represent streets on an abstract level as continuous lines and crossings as decision points, such a simplistic model implies severe limitations for pedestrians. The suggested routes will often direct the user on a detour that strictly follows the modeled path segments, even if it would be possible for a pedestrian to abbreviate the route across large places. In (Stahl, 2004) we argued that the route planning component should reflect this and provide shortest paths within arbitrary spaces. This requires new algorithms and the modeling of the actual shapes of streets and sidewalks rather than abstracted centerlines. Gaisbauer and Frank (2008) share our opinion and claim that a model for pedestrian navigation should consider *decision scenes* that comprise the local vista space around a decision point.

Besides graphical route depictions, such as maps, often verbose directions are preferred. In this case we have to use a human-readable vocabulary instead of coordinates in order to denote locations. In the case of public information systems, e.g. at an airport, multiple languages have to be supported. It is not only for verbalization, but for a meaningful expression of location in general that we propose terminological knowledge to refer to physical or logical entities like rooms, floors, and wings of a building. Such a symbolic (or qualitative) location model is further useful as an index from which we can infer the overall context influencing the mobile application (Dix, et al., 2000). For user assistance in ubiquitous computing environments, we wish to express the state of the infrastructure and the locations of users and devices in one coherent database. In particular, qualitative spatial relations of containment and neighborhood are useful for range queries about the surrounding space.

In conclusion we need both, a geometrical and a symbolic (or qualitative) location model, with strong interrelationship.

2.4.2 Coordinate Systems

A ***coordinate*** *x* is an identifier which specifies the position of a located object or location with respect to a given **coordinate system**. In geographical applications, coordinates often denote geographical objects by abstract symbolic identifiers. For example, a postal address is a coordinate that represents a certain building as a tuple made of a zip code, street name, and number. Such a naming schema can be considered as a **symbolic coordinate system**. Whereas in mathematics, a **geometric coordinate system** assigns a tuple of real numbers to each point in an n-dimensional space. Geometric coordinate systems can be based on Cartesian or spherical coordinate systems. A **geographic coordinate system** expresses every location on Earth by two of the three coordinates of a spherical coordinate system, called Latitude and Longitude. Details about these coordinate systems can be found in Appendix D.

2.4.3 Geometric Location Models

Geometric location models represent the position or actual shape of geographical objects and buildings by a set of numerical coordinate tuples. One benefit of this representation is that it allows the calculation of Euclidean distances between any two points or objects or the length of paths. If coordinates are conceived as nodes of a graph, more sophisticated algorithms can be applied, such as the A* search for shortest paths in graph networks. Geometric models can also be visualized from arbitrary viewpoints through standard methods of computer graphics.

Typical datastructures are polylines and polygons; a polyline represents a continuous line as a series of connected line segments, whereas a **polygon** represents a closed line or surface by its vertices. Usually, vertices are given as two or three dimensional coordinates, but also two and a half dimensional models are possible. In this case, each polygon object has an additional attribute, which extrudes the outline along the vertical axis. Thus buildings can be easily represented by their cross section and height.

Road map data for car navigation systems, as provided by *Open Street Map*[12] for example,

[12] http://www.openstreetmap.de/

represents the centerline of streets as continuous line segments, which are annotated by additional symbolic information such as the number of lanes, house number ranges, and points of interest. For pedestrian navigation purposes, polygon representations of the actual shape of streets and places are more appropriate. The *Geography Markup Language 2.0* (GML, 2001) by *Open-GIS* is an XML encoding for the transport and storage of geometrical location models that represent the spatial properties of geographic features in two dimensions. *VRML* (Virtual Reality Modeling Language)(Carey and Bell, 1997) by *Web3D* is a interchange format for 3-D scenes, which defines elements such as hierarchical transformations, light sources, viewpoints, geometry, animation, material properties, and texture mapping. Geometry can be represented by coordinate sets as described above using the *IndexedFaceSet*, *IndexedLineSet* and *Extrusion* nodes. The successor *X3D* is based on XML.

2.4.4 Symbolic Location Models

A *symbolic location model* represents locations by abstract symbols or names, and can deliver meaningful information about places and their relations. Meaningful symbolic names for locations are often required in addition to a geometric model for human-computer interaction, since coordinates themselves are not human readable and only useful for visualization, but not for verbalization. Symbolic location identifiers are also useful for rule-based inference systems in artificial intelligence or predicate logics. On the other hand, the distance between symbolic coordinates is not defined, so that information about containment and connectedness of locations has to be represented through additional topological relations.

It is typical for symbolic location models that their elements are hierarchically organized, since geographical categories, such as continents, states and cities, imply a natural order of containment. Leonhardt (1998) suggests the *location domain model*, which hierarchically represents geographical areas in a graph of domains. The graph forms a lattice instead of a tree structure, which allows each node to have multiple parents. For example, a room can be contained in a wing-domain, and at the same time the room can also be contained in a floor-domain. Both domains themselves are contained in the building domain.

Kray et al. (2008) suggest a symbolic location model for indoor navigation support through public displays that incorporates three types of relations: directional, connectional, and mereological relations. Directional relations include three pairs: *<north-of>/<south/of>, <west-of>/<east-of>*, and *<on-top-of>/<below>*. Six connectional relations define how two locations are connected to each other: *<directly-connected>, <connected-by-door>, <connected-by-stairs>, <connected-by-escalator>*, and *<connected-by-elevator>*. The containment relation *<contains>* defines hierarchical areas.

From an Artificial Intelligence perspective it is an obvious approach to base a symbolic location model on an ontology. According to Gruber (1993)

- An **ontology** is an engineering artifact, which constitutes a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary. It provides a shared understanding and formal and machine readable model.

We will now see two examples for symbolic location models, which are both based on ontologies that represent a vocabulary of geographic names and places.

2.4.4.1 *The Getty Thesaurus of Geographic Names*

The *Getty Thesaurus of Geographic Names Online* [13] (TGN) contains the current and historical names of more than one million inhabited places together with encyclopedically information about them. The TGN includes all continents and nations of the modern political world, as well as historical places. It includes physical geographical features, such as mountains and rivers, and administrative entities, such as cities and nations. The TGN implements a hierarchical relation between its entities, such as continents, states, and inhabited places. For example, a search for the city of Saarbrücken returns besides a detailed description of the place the following location path "*Top of TGN (root) > World > Europe (continent) > Germany (nation) > Saarland (state) > Saarbrücken (inhabited place)*". The emphasis in TGN is on places that are culturally important for art and architecture, and the model is limited to the level of inhabited places. So it is a good example for a symbolic location model in general, but not suitable for user assistance systems, which require a finer grained model.

2.4.4.2 *The UbisWorld Ontology*

The *UbisWorld* model has been developed by Heckmann (Heckmann, 2006) to represent some parts of the real world like an office, a shop, a museum or an airport, for ubiquitous computing applications. **Situational Statements** (Heckmann, 2003) describe the state of the world in sentences made of a subject, predicate, and object, which represent the properties and relations between persons, things, and locations. The vocabulary of these sentences is provided by an underlying ontology, the *UbisOntology*, which defines the necessary classes and predicates, such as a taxonomy of physical things, user characteristics, spatial elements, and relations.

The **UbisEarth** ontology currently defines about 18 million geographic locations, which are symbolically represented at different levels of granularity, like *continent*, *country*, *city*, and *street*. It is also possible to express spatial topological relations like inclusion and neighbourhood between spatial elements. In addition, roles like *country* are used to list all countries within a continent, see Figure 15 for example. The hierarchy begins with the Earth element, which has several instances of continents, like Europe. The *continent* of Europe is parent to many countries, which have been graphically abbreviated here in order to reduce the size of the depiction. The *country* of Germany comprises several *regions*, including Saarland, which leads to the *city* and streets of Saarbrücken. Whereas other symbolic geographic models, such as the TGN or postal addresses, typically end at the city or street level, the UbisEarth ontology provides finer levels of detail. The granularity levels *building*, *room*, *shelf*, and *board* allow for a detailed modeling of indoor spaces and storage elements. For example in Saarbrücken, the ontology includes some *building* instances, like "D3 1", that includes several *rooms*. Finally, some shelves and their boards are listed as *part-of* "room C0.16" and can be used as precise symbolic identifiers to denote the location of items within an environment, such as books in a library or products in a store.

The concept of situational statements provides for the flexibility that is needed to model any situational context that might be relevant to a specific ubiquitous computing application. In particular it is possible to model various properties of spatial elements, such as room-

[13] http://www.getty.edu/research/conducting_research/vocabularies/tgn/index.html visited 23.07.2009

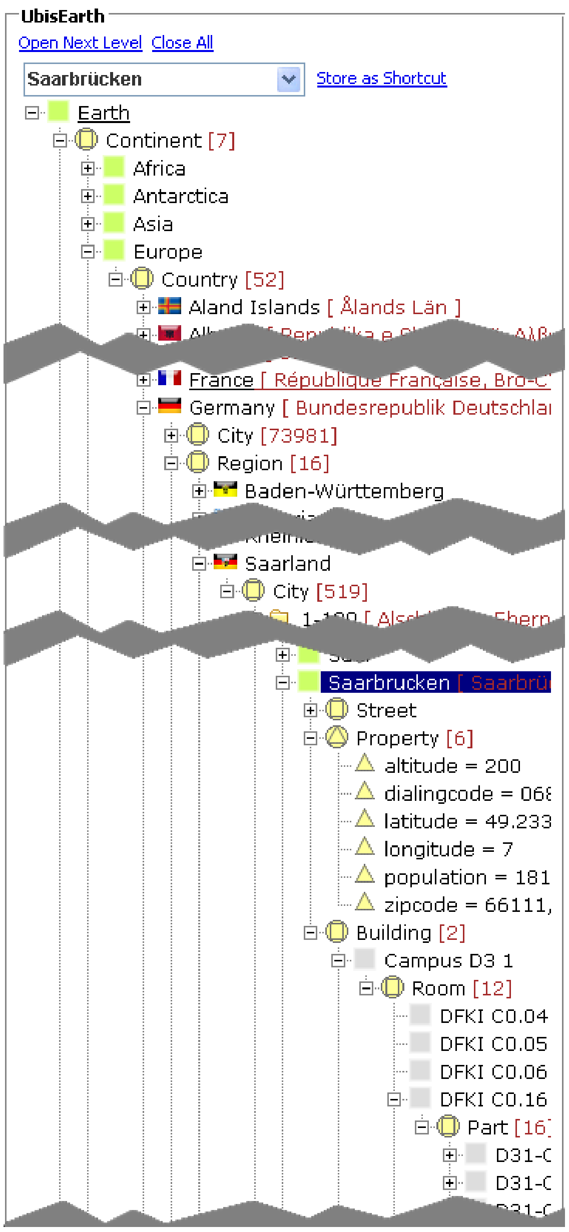
temperature, noise-level or humidity. The default ontology can easily be extended to specific domains. Additionally, the presence of persons and physical objects can also be expressed and visualized. Figure 16 shows the current situational context of *room 1.24* in *building 36* at *Saarland University*.

UbisWorld provides a convenient Web-based user interface, which allows to inspect the ontology through trees as shown in Figure 15 and Figure 16. Several export modules allow for representing query results in various Semantic Web (Fensel et al., 2003) languages, such as the Resource-Description Framework (RDF) (Lassila, 1999; RFC 2396; Manola et al., 2004). The RDF language has been originally designed for the semantic annotation of Web content through the use of ontologies. It allows the computer to understand the meaning of textual and graphical information found on the website and helps to improve query-based search results beyond purely syntactic keyword matching. However, ontologies are also useful to interchange knowledge between multiple systems as it is described later in this thesis in Section 4.1.4 on Ubiquitous User Modeling.

Internationalization of identification names, for cities for instance, bears two major problems: which language to choose and which character set to choose for those characters that are not covered by the ASCII character set. One could choose the user's mother language, an interlingua like English, or the language of the country where the city is located. However, the following examples given by Heckmann and Loskyll (2009) indicate that there is no solution that fits all:

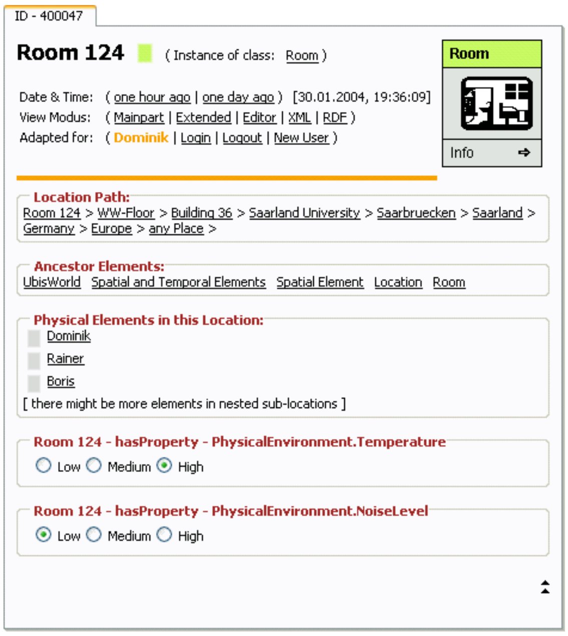
- **The city of Prague.** Which should be the label of the concept for the capital in the Czech Republic for a German user? The German name is Prag; the English name is Prague; while the original Czech name is Praha;
- **The city of Munich**. Which should the label of the concept for the capital of Bavaria be for a Czech user? The German name is München; however, the city is often spelled Muenchen since the usage of the non-ASCII character 'ü' could lead to problems; the English name is Munich.

UbisWorld supports internationalization and localization through the encoding of multiple labels for any concept into one *UbisLabel*. This approach allows the adaptive selection and presentation of a label according to a situation-aware strategy. Heckmann and Loskyll (2009) have further developed the *UnderscoreEncoding* representation for UbisLabels in order to support i) readability for humans, ii) stability even if the resource is moved, iii) efficiency regarding the use of labels in databases/URIs/RDF, and iv) informativeness. This encoding has been developed in order to represent special characters in an efficient manner without using any characters apart from alphanumerical letters plus the underscore. Multilingual names are represented by a single string of ASCII characters that contains all languages and conforms to the syntactical limitations of URIs and other semantic web languages, such as OWL or RDF.



```
UbisEarth
Open Next Level  Close All
Saarbrücken                 Store as Shortcut
⊟ Earth
  ⊟ Continent [7]
    ⊞ Africa
    ⊞ Antarctica
    ⊞ Asia
    ⊟ Europe
      ⊟ Country [52]
        ⊞ Aland Islands [ Ålands Län ]
        ⊞ France [ République Française, Bro-C
        ⊟ Germany [ Bundesrepublik Deutschla
          ⊞ City [73981]
          ⊟ Region [16]
            ⊞ Baden-Württemberg
            ⊟ Saarland
              ⊟ City [519]
                ⊟ Saarbrucken [ Saarbrü
                  ⊞ Street
                  ⊟ Property [6]
                      altitude = 200
                      dialingcode = 068
                      latitude = 49.233
                      longitude = 7
                      population = 181
                      zipcode = 66111,
                  ⊟ Building [2]
                    ⊟ Campus D3 1
                      ⊟ Room [12]
                          DFKI C0.04
                          DFKI C0.05
                          DFKI C0.06
                        ⊟ DFKI C0.16
                          ⊟ Part [16]
                            ⊞ D31-C
                            ⊞ D31-C
```

Figure 15: Browsing geographic elements online in the *UbisWorld* Web interface.



ID - 400047

Room 124 🟩 (Instance of class: Room)

Date & Time: (one hour ago | one day ago) [30.01.2004, 19:36:09]
View Modus: (Mainpart | Extended | Editor | XML | RDF)
Adapted for: (Dominik | Login | Logout | New User)

Room
Info ➩

Location Path:
Room 124 > WW-Floor > Building 36 > Saarland University > Saarbruecken > Saarland > Germany > Europe > any Place >

Ancestor Elements:
UbisWorld Spatial and Temporal Elements Spatial Element Location Room

Physical Elements in this Location:
Dominik
Rainer
Boris
[there might be more elements in nested sub-locations]

Room 124 - hasProperty - PhysicalEnvironment.Temperature
○ Low ○ Medium [✓ High]

Room 124 - hasProperty - PhysicalEnvironment.NoiseLevel
[✓ Low] ○ Medium ○ High

Figure 16: Properties of a room instance, as represented online in *UbisWorld*.

Using *UnderscoreEncoding*, the city of Munich can be labeled as follows:

(1) Munich
(2) EN..Munich
(3) EN..Munich...FR..Munich...DE..M_AA0nchen
(4) CITY....2867707.ENFR..Munich...DE..M_AA0nchen

Label (1) and (2) represent only the English name, whereas Label (3) also encodes the French name, which is actually the same, and the German name “München”. The umlaut “ü” is represented by the character sequence “_AA0”. The UbisName (4) results from the concatenation of the UbisPointer and the UbisLabel with the 4-dot-operator.

2.4.5 Hybrid Location Models

In order to represent both the logical structure and the physical dimension of the environment, the symbolic model can be merged with geometric information on the spatial elements. The combination of a geometric and symbolic model is called a *semi-symbolic* or *hybrid* model (Leonhardt, 1998). In such a model, the coordinates and shapes of a geometric model are mapped to the abstract identifiers of the symbolic model. Thus it is possible to implement functions which map geometric coordinates to symbolic names, and vice versa. For example, car navigation services require geo-coding functions that map postal addresses into geographic (Longitude and Latitude) coordinates for route finding.

Another speciality of hybrid location models is the possibility to embed multiple local geometric coordinate systems, which can be considered as subspaces. The *aura location indentifier* ALI (Jiang and Steenkiste, 2002) implements a service to handle spatial queries for ubiquitous computing. The aura location model allows each space in the symbolic location hierarchy to define its own coordinate system with limited scope that can be used to define points or areas within that subspace. Different spaces may use different coordinate systems; on top of the hierarchy, a global coordinate system such as WGS84 can be used to map cities and buildings, whereas on a finer level of granularity, rooms can define their own subspace using a local Cartesian coordinate system. Dürr and Rothermel (2003) present a similar hybrid location model for fine-grained geocasting, with embedded local reference systems.

In (Stahl and Heckmann, 2004b) we discuss how Semantic Web languages like RDF can be used for hybrid location modeling. We identify the symbolic locations and also the geometrical models as Web resources, and represent their relationships using RDF statements, as shown in Figure 17. The geometry is represented on the left hand side in two granularities, an abstract campus map (above) and a detailed floor model (below) that uses its own local coordinate system. On the right hand side the according symbolic instances are shown that represent the campus, the buildings, and the rooms. The binary relations <*Geometric_model*> and <*Symbolic_model*> link the geometric instances with their symbolic representation, and vice versa. These links allow the implementation of mapping functions.

A typical use case of such a hybrid model would be to give the user a verbose description of their location, based on geometric coordinates that are given by a positioning system. First, the system has to look up the geometric model and perform algebraic tests which region contains the user's location. Then the system can use the reference to the symbolic model to find out qualitative information about that region, such as multilingual labels and containment information. For example, the system can inform the user that he/she is currently located in the kitchen, which is situated in the second floor of building *E1 3* at the campus of Saarland University, Saarbrücken. Vice versa, if the user specifies a navigational destination, the system can look up the symbolic model for this place. Then the system can follow the link to the geometric model to figure out the coordinates of the destination and calculate a route.

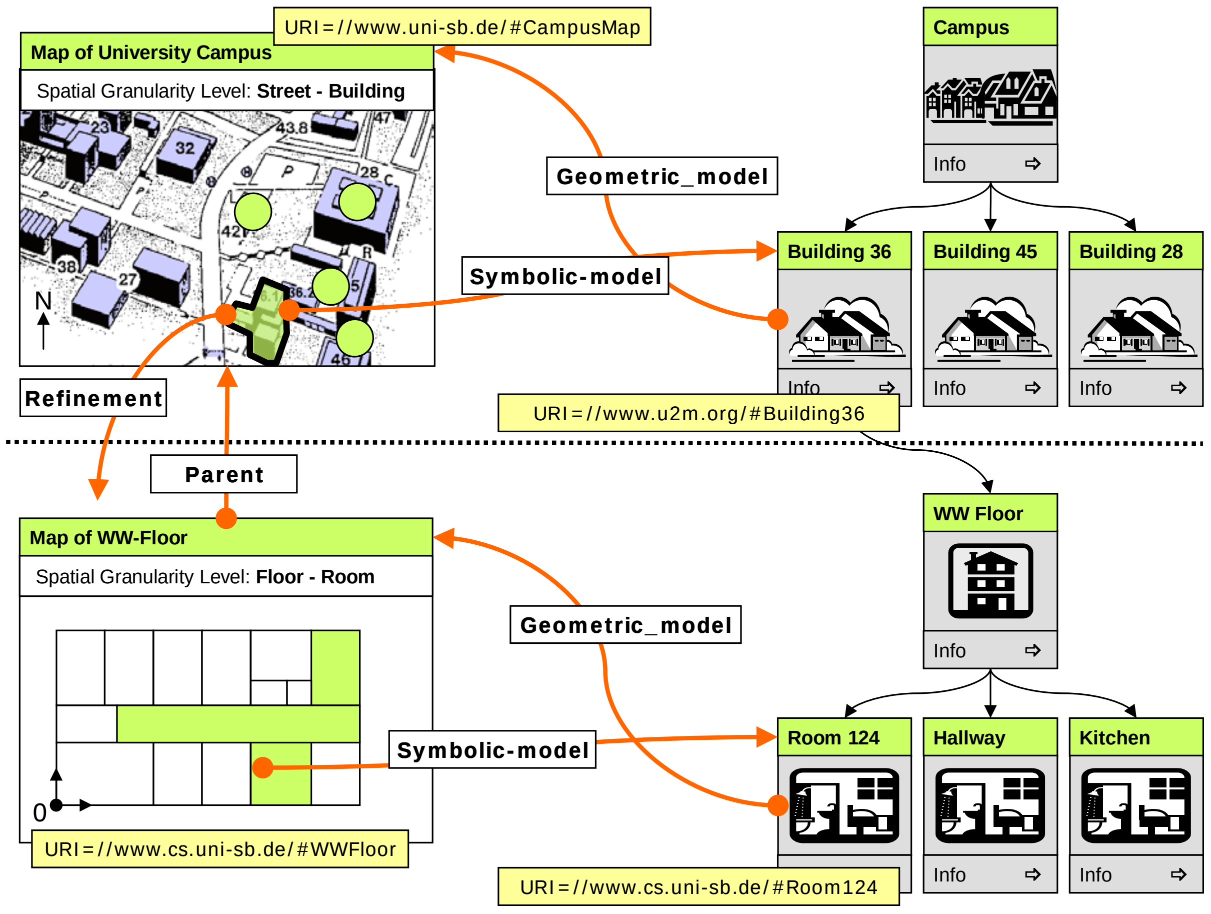

URI=//www.uni-sb.de/#CampusMap
Map of University Campus
Spatial Granularity Level: **Street - Building**
Geometric_model
Symbolic-model
Refinement
Parent
Campus
Info
Building 36
Building 45
Building 28
Info
URI=//www.u2m.org/#Building36
Map of WW-Floor
Spatial Granularity Level: **Floor - Room**
Geometric_model
Symbolic-model
URI=//www.cs.uni-sb.de/#WWFloor
WW Floor
Info
Room 124
Hallway
Kitchen
Info
URI=//www.cs.uni-sb.de/#Room124

Figure 17: Integration of geometric (left) and symbolic (right) models into a hybrid location model by URIs.

The symbolic model can also be used to represent logical entities which have no counterpart in the geometric model, for example the floor, or wing of a building. Such hierarchical structures can easily be modeled in the symbolic model through containment relations. Connectional relations, such as the set of relations suggested by Kray et al. (2008), can be useful to plan and describe routes through a building on a symbolic level. Whereas the geometric model is highly suitable to figure out the shortest path, other navigational tasks, such as finding the simplest or most interesting route, require qualitative information from the symbolic model. Even if a shortest path is desired, such a symbolic pre-computation can help to drastically reduce the geometric search space.

Furthermore, the symbolic model can be used to represent all kind of domain knowledge for ubiquitous computing applications, e.g. who works in an office, what are the opening times of a store, and which appliances are installed in the kitchen. Based on such additional knowledge, navigational assistance systems can be implemented that allow their users to directly formulate their goal, e.g. “take me to the office of Mrs. Borchers”.

2.5 Summary

In this chapter we presented the basic concepts that will be used later in this thesis. We first introduced fundamental concepts of interaction design such as conceptual models based on objects, metaphors, and activities. We then presented a survey of computing paradigms besides the desktop computer such as Ubiquitous Computing and Instrumented Environments. In this context, we also gave a brief overview over an interactive development process model for interaction design that includes prototyping and evaluation.

Since this thesis deals with an activity-based design approach towards user assistance, we have introduced the basic concepts and paradigms of Activity Theory according to Leontiev. In addition, we have motivated the use of Activity Theory for interaction design and presented the related theories Activity-Centered Design, taskonomies, and Activity-Based Computing.

We assume that every activity includes navigational sub-goals that require assistance. In this context we have reviewed the concepts of wayfinding and hierarchical decision plans. Further aspects concern the representation of spatial knowledge, such as route and survey maps, and landmarks. We have also discussed decision points as a measure for the complexity of wayfinding problems.

In addition to wayfinding we have introduced location models for Ubiquitous Computing, beginning with a discussion of the requirements for a location model space. On the one hand we have presented geometric coordinate systems and location models. On the other hand we introduced symbolic location models and examples thereof, particularly the UbisWorld, which will be used throughout this thesis. Finally, we have shown how both models can be merged to a hybrid location model and explained the resulting advantages for ubiquitous computing applications.

3 Related work

This thesis aims to develop a structured method and tool support for the modeling and designing of instrumented environments. This comprises the modeling of the user's needs and the instrumentation of the environment. In this chapter we first present some examples of intelligent environments from home, retail, and industrial domains and we highlight some assistance applications for everyday life situations. Then we will introduce related work on the modeling of tasks and activities, and present model-based design methods. Finally, we give a survey of tools that cover various aspects of geometric location modeling and discuss their abilities with respect to our goals.

3.1 Survey of Intelligent Environments

In this section we present examples for intelligent environments that assist i) users in everyday live situations at home, including care taking of elderly persons, ii) customers in their shopping tasks in retail environments, and iii) service staff in industrial plants.

3.1.1 Smart Homes and Care

3.1.1.1 *Project Oxygen, MIT, USA*

The name of the project Oxygen[14] refers to the air we breathe; oxygen is necessary for us, yet invisible. The project assumes that future computing will be like air—ubiquitous and freely available for us. The $50 million five-year project started in 1999 with the goal to make computing more human-centered and user-friendly, and to show how computing can help to assist average people, not specialists, in their everyday tasks. The computer should adapt to its users, not vice versa. Human-centered computation aims to support social interaction in its familiar context. People communicate with computation easily, as they do with each other, using shared knowledge and intelligence. Freely-available computation makes it easy for people to monitor and control their environment. Physical objects become immensely more useful and usable, and people and agents in other environments become immensely easier to reach.

To support highly dynamic and varied human activities, the Oxygen system must master many technical challenges to be *pervasive*, *embedded*, *adaptable*, and *powerful*. Furthermore, the system should be *intentional*—it must enable people to name services and software objects by intent, for example, "the nearest printer," as opposed to by address; the system should also be *eternal*—it must never shut down or reboot.

Since there were no adequate mobile devices on the market at the beginning of the project,

[14] http://oxygen.lcs.mit.edu/

the group focused part of its work on the development of energy-efficient processors, ad-hoc network protocols, and computing infrastructures. As a result, handheld devices called Handy21s (H21s) are able to adapt to their user, location, and network bandwidth and can dynamically download program code for various applications, for example a Web radio. Computational devices, called Enviro21s (E21s), were developed to be embedded in homes, offices, and cars in order to sense and affect their immediate environment. E21s provide a local-area computational and communication back-plane for *intelligent spaces*; therefore E21s are connected to nearby sensors, actuators, and appliances, suitably encapsulated in physical objects. An E21 might control an array of microphones, which Oxygen's perceptual resources use to improve communication with speakers by filtering out background noise. E21s provide sufficient computational power throughout the environment to i) communicate with people using natural perceptual resources, such as speech and vision, ii) to support Oxygen's user technologies wherever people may be, and iii) to monitor and control their environment.

Another outcome of the project was the Cricket location support system; one of the first indoor-positioning systems. It is based on beacons, mounted on walls or ceilings, which transmit ultrasound and RF signals; compact listeners attached to mobile devices use the difference in signal arrival times to determine their position.

The *intelligent Conference Room* (AIRE Group[15], MIT) implements an intelligent space. It is a highly interactive environment that uses embedded computation to observe and participate in normal, everyday events. Ceiling-mounted microphone and camera arrays enable it to listen to people and observe what they do, as illustrated in Figure 18. People can speak with, gesture to, and interact with it in other complex ways. The intelligent room has been fitted as a conference room, as depicted in Figure 19; its features include large whiteboard wall surfaces which double as four integrated projection displays, speech input through hand-held close-talk microphones, an LED display for weather and news reports, a five-speaker sound system that can `rotate' the sound focus to any wall, and two steerable and two stereo cameras for detecting and tracking people. A wall-mounted LCD touchscreen is used for room configuration and status displays.

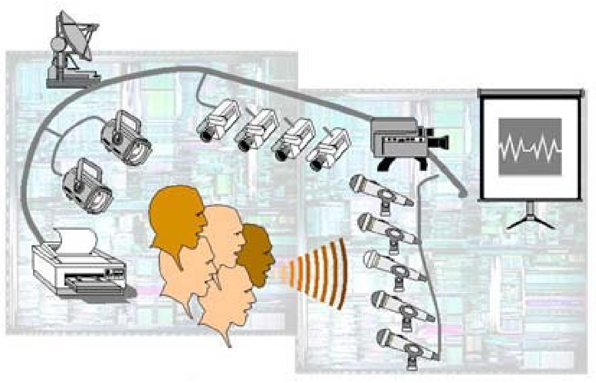


Figure 18: Camera and microphone array. Image: MIT.

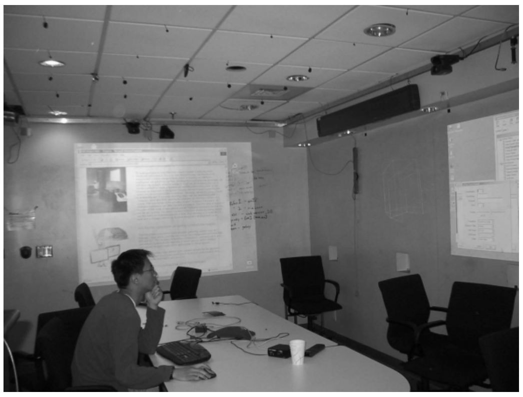


Figure 19: The Intelligent Room. Image: MIT.

[15] http://aire.csail.mit.edu/

3.1.1.2 *Aware Home, Georgia Tech, USA*

The Aware Home Research Initiative[16] (AHRI) is an interdisciplinary research endeavor at Georgia Tech aimed at addressing the fundamental technical, design, and social challenges presented by the following research questions:

- Is it possible to create a home environment that is aware of its occupants' whereabouts and activities?
- If we build such a home, how can it provide services to its residents that enhance their quality of life or help them to maintain independence as they age?

The Aware Home Residential Laboratory is a three-story, 468m^2 home that functions as a living laboratory for interdisciplinary design, development, and evaluation. It has been opened in 2000 for technology- and human-centered research in ubiquitous computing for everyday activities. The house is fully functional, so that people could permanently live there in an authentic setting, as shown in Figure 20. Two independent living spaces allow for controlled experiments with inhabitants living on one floor while prototyping or providing demonstrations on the other floor. Each space consists of two bedrooms, two bathrooms, one office, kitchen, dining room, living room, and laundry room. In addition, there is a shared basement with a home entertainment area and control room for centralized computing services.

Current research in the Aware Home is focused on three areas: i) Future tools for the home, ii) Chronic care management in the home, and iii) Digital entertainment and media.

The first area investigates new opportunities to support tasks in the home based on embedded computing, sensing, and actuation technologies, coupled with new infrastructure in the built environment itself. Research is targeted on the technology for new tools as well as the domestic setting in which they will be used. Current applications are for example an energy consumption display that helps inhabitants to conserve energy, a message center (Figure 21), and a family telepresence system that helps people to stay in touch through natural activities, like playing games, reading together, and doing homework.

The second area aims to develop support systems for adults who have to care for both their children and their aging parents. Similar to the Ambient Assisted Living initiative, the goals of this area are to support coordination of tasks between formal and informal caregivers, and to reduce health care costs by allowing people to live independently in their own homes. The project has developed Interactive tools that promote health education and care by utilizing sophisticated home health monitoring. Other applications enhance portraits of family members with ambient information about their well-being, or help visually imparted people to locate lost items in their home.

Finally, the project explores new opportunities for entertainment in leisure time and is concerned with usability issues and privacy concerns of media sharing.

[16] http://awarehome.imtc.gatech.edu/

Figure 20: Inside the Aware Home.
Image: Georgia Institute of Technology.

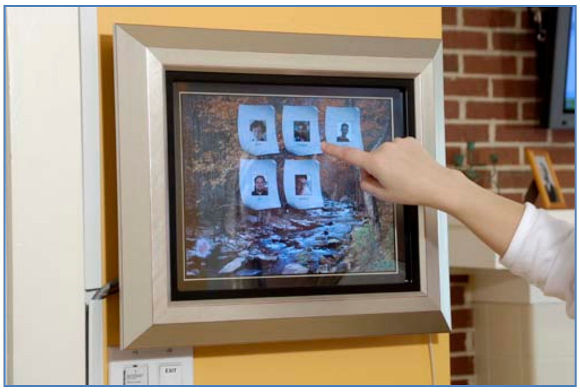


Figure 21: Message Center for the Family.
Image: Georgia Institute of Technology.

3.1.1.3 *ExperienceLab, Philips High Tech Campus, Eindhoven, NL*

Whereas the former environments have a strong interest in pervasive computing technologies, such as embedded system architectures and indoor positioning, the ExperienceLab has been built by Philips Research Europe in order to develop novel interaction concepts that allow the user to communicate with their electronic environment in a natural way. According to de Ruyter (2007b), Philips follows a user-centered development process that goes beyond the development of scenarios and the translation of use cases into system requirements. The ExperienceLab has been built to expose users to experience prototypes, which are implemented by the researchers in order to evaluate the usability of novel interaction concepts. In order to test the systems in a natural environment, Philips has built three different labs that closely resemble a home-, shop-, and care-environment. All of them provide an observation infrastructure that allows the researchers to capture and analyze the interaction between the users and the prototype systems.

3.1.1.4 *HomeLab, Philips*

The HomeLab[17] has been built as a two-storey house with a living room, a kitchen, two bedrooms, a bathroom, and a study, and has been opened on April 24, 2002. It allows researchers to evaluate prototypes of intelligent technology through target users in a fully functional home environment. The lab provides video observation equipment to record and analyze user's behavior in order to better understand their needs and motivations.

Hartson explains the user-centered design approach of the HomeLab as follows:

> *"The user's context for technology has two parts: the physical, cultural and social aspects of daily living activities on the one hand and the interface to technology on the other. Philips' user-centered engineering is what bridges that gap; translating from the user's cognitive and perceptual language to the physical language of the controls and displays of technology". (de Ruyter, 2003:p26)*

According to de Ruyter (2003), Philips has an interest in the development of interactive

[17] http://www.research.philips.com/technologies/projects/homelab/index.html

systems that are both supportive and entertaining, but also interwoven into the social context at home. As broadband internet becomes widely available, researches are interested in applications that meet the users' needs. Hence Philips has investigated how to support the sharing of content and experiences and how to create the feeling of being together.

Furthermore, research has been focused on how to make home entertaining systems more exciting and relaxing. One prototype has already become a well known feature of Philips flat screens today; the Living Light system (de Ruyter, 2003:15) comprises five so-called LightSpeakers which illuminate the living room in different colors (Figure 22), according to the movie that is playing on the TV screen. The Philips AmbiLight[18] product has been reduced to one light source behind the TV which provides ambient background-lighting effects that match the colors of the images shown on the TV.

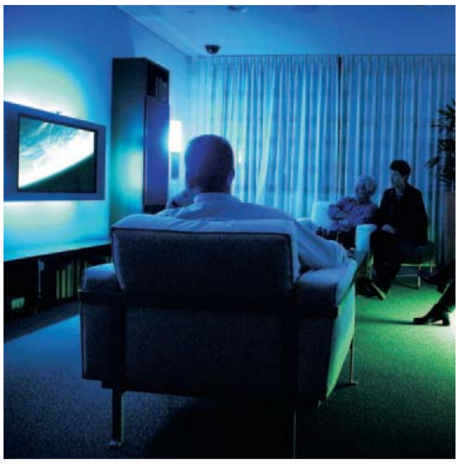


Figure 22: Living Light. Image: Philips.

Figure 23: Seniors in the CareLab. Image: Philips.

3.1.1.5 *CareLab, Philips*

Research in the HomeLab is complemented by a lab for seniors, the CareLab (see Figure 23). This apartment is located at the Philips High Tech Campus in Eindhoven and comprises one bedroom, bathroom, and living room with a kitchenette. It is used as an instrument for ensuring early user involvement in the development of innovative applications of Ambient Assisted-Living technology.

According to de Ruyter (2007), research in the CareLab is focused on three applications: i) a lifestyle assistant, ii) cognitive stimulation, and iii) social connectedness. The former application targets the seniors need for safety and protection at home. A range of distri-buted sensors register activities, which are processed by a context-aware reasoning engine. This reasoning engine identifies potential critical incidents and in response alerts care centers or relatives. Cognitive Stimulation has been implemented as an IP-TV based service that helps the elderly to train their mental abilities through games. Social Connectedness is realized through awareness systems that help individuals or groups build and maintain a peripheral awareness of each other. The system uses a sensor network to capture contextual information related to the daily activities of elderly people and presents automated diaries on a picture frame as awareness information for their children living at a remote location.

[18] http://www2.philips.de/entspannter_fernsehen/

3.1.1.6 *Assisted Living Laboratory, Fraunhofer IESE, Kaiserslautern*

The Assisted Living Laboratory[19] (ALL) is located at the Fraunhofer IESE in Kaiserslautern. Similar to the other labs presented above, it consists of entrance, living room, sleeping room, bathroom, and kitchen. However, in contrast to other labs the bathroom and kitchen facilities are not functional, so that people can not actually live there. The AAL has been designed as platform for the demonstration and evaluation of application prototypes that allows for usability studies in a realistic environment.

The research in the ALL is conducted in conjunction with several projects; the project BelAmI[20], the EU-project EMERGE[21], and the BMBF-project AmbiComp[22]. The goal is the user-centered development of emergency assistance systems in intelligent environments. The system tries to recognize critical situations from sensor data and to initialize adequate assistance through care takers or an emergency doctor.

The AAL has been instrumented with a large variety of ambient sensors in order to recognize unusual situations from context. Lights, switches, doors, and window blinds are automated using a KNX-based bus. Power sockets have been instrumented and report usage of electric appliances. Multiple systems allow for indoor positioning, e.g. a smart carpet equipped with RFID tags, and RFID/ultrasonic beacons mounted in the ceiling. Intelligent household appliances recognize their state, and an intelligent walking stick recognizes falls. Vital data monitoring devices collect basic vital data, such as pulse and skin temperature. Various audio and visual devices in several rooms allow for multimodal interaction and video telephony communication services. The AAL environment also includes a service robot in form of a small table which can be used to transport items between rooms.

Sensor data is evaluated by a rule-based activity recognition framework (see Storf et al., 2009) in order to distinguish regular everyday activity from emergency situations that require assistance. The framework will be explained in more detail in Section 4.2.3.1. The system assumes a single-user scenario, since such persons are more likely to require assistance through intelligent environments than couples. One of the usability goals is to avoid any instrumentation of the user; people should not be required to wear any tags or devices with them. All necessary data should be collected through the environment's sensors.

3.1.1.7 *BAALL, DFKI, Bremen*

The Bremen Ambient Assisted Living Lab (BAALL)[23] is an apartment suitable for the elderly and people with physical or cognitive impairments. With a size of 60 m², it comprises all necessary conditions for trial living, intended for two persons, as shown in Figure 24. It is situated at the Bremen site of the German Research Center for Artificial Intelligence (DFKI) in one of the labs at the research department. The goal of the project is to investigate how the

[19] Assisted Living Laboratory, http://www.aal.iese.fraunhofer.de
[20] Bilateral German-Hungarian Collaboration Project on Ambient Intelligence, http://www.belami-project.eu
[21] EMERGE, Emergency Monitoring and Prevention, http://www.emerge-project.eu
[22]AmbiComp, Software Entwicklung mit universell kombinierbaren AmI Komponenten, http://www.ambicomp.de
[23] http://www.baall.net

living environments of seniors to-be can be instrumented with infrastructures that allow incremental upgrades with user assistance systems, as required in their future.

One particular focus of BAALL is to compensate physical impairments of the user through mobility assistance and adaptable furniture. Having a strong research background in robotics, the DFKI Bremen has developed the *Bremen Intelligent Wheelchair* and *Intelligent Walker* which are fitted with laser range scanners for indoor positioning and assist their user to safely navigate within the lab. The lab has also been equipped with electronic sliding doors that will open automatically to let the wheelchair pass through. In order to allow a wheelchair-dependent user to stay together with their non-impaired partner in the same living environment, user-adaptable furniture has been installed and can be adjusted to fit the different physical requirements. The washbasin and the kitchenette can be electronically raised and lowered, and the bed has a mattress with five individually adjustable sections that allow a variety of configurations.

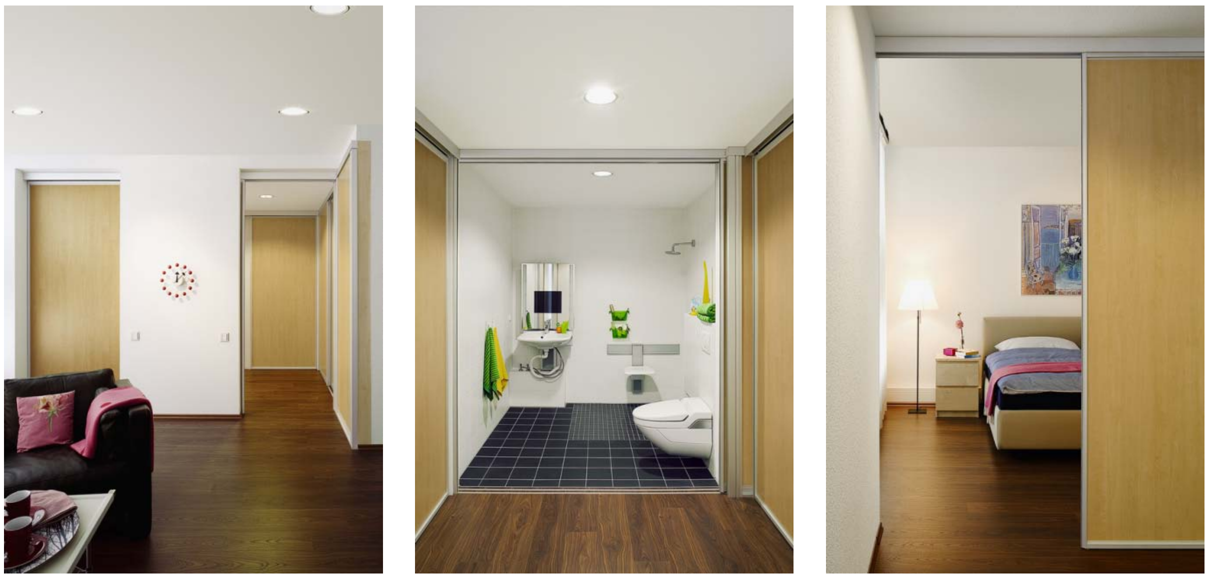


Figure 24: The Bremen Ambient Assisted Living Lab is a 60 m² apartment that is suitable for elderly people with physical or cognitive impairments. Images: ©raumplus.

3.1.1.8 *IDEAAL, OFFIS, Oldenburg*

The IDEAAL apartment for seniors has been opened in October 2005 as one of the first European AAL labs for research and demonstration purposes. It has been renovated in 2008 and enhanced to a fully functional 48m² flat with living room, bedroom, kitchen, and bath. The realization and furnishing of the IDEAAL senior apartment has been oriented on the preferences of the target user group. The apartment serves as a demonstration facility for realistic AAL scenarios, in which technology supports independent living and more efficient care-taking of seniors. It allows for both feasibility and usability studies in a domestic environment. The developed prototypes are presented to medical experts as well as end-users.

3.1.1.9 *iSpace, University of Essex, Colchester, UK*

The intelligent dormitory 2 (iSpace)[24] is a two bedroom flat at the University of Essex that has been designed for research into intelligent buildings and adaptive environments. It offers the possibility to deploy and examine embedded agents that learn from the user within intelligent environments. The apartment provides specially constructed walls to hide the infrastructure of networked sensors and actuators for ubiquitous computing.

3.1.1.10 *Ocha House, Ochanomizu University, Tokyo*

The Ocha House[25] is an in-campus experimental one-floor house of 90 m² (Figure 26) that was constructed in March 2009 in Koishikawa area at Ochanomizu University, one of two national woman's universities. Current work around Ocha House is focused on assistive domestic environments and awareness support between families and couples that live in separate homes.

The *SyncDecor* (Tsujita et al., 2008) project pairs common, day-to-day appliances, such as desk lamps, and remotely synchronizes them in order to communicate everyday actions between couples while maintaining a sustained and natural usage pattern. *SyncDecor* aims to provide awareness or cognizance between partners, thereby creating a virtual feeling of living together. The *TagTansu* (Tsukada et al., 2008) wardrobe has been instrumented with a digital camera to capture pictures of clothes and to share them on social fashion networks.

The *Smart Makeup Mirror* (Iwabuchi et al., 2008) is an electronic dressing table that aids woman in wearing makeup easily and make the process enjoyable (Figure 25). The system uses a high-resolution camera above a display to facilitate the process of applying makeup through an electronic mirror. The mirror recognizes makeup tools through optical marker tracking and a proximity sensor and automatically enlarges parts of the face where makeup is applied. The mirror can also simulate different lighting conditions, and allows its users to discuss the makeup results online with friends.

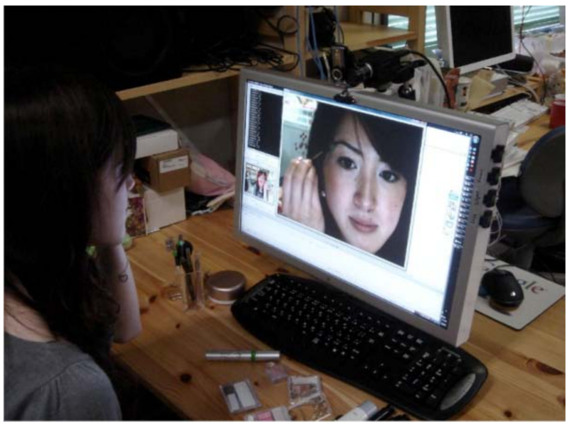


Figure 25: Smart makeup mirror

Figure 26: Ocha house is built on a skeleton of laminated wood frames.

[24] http://iieg.essex.ac.uk/idorm2/index.htm

[25] http://ochahouse.com/

3.1.2 Intelligent Retail Environments

3.1.2.1 *FutureStore, Metro Group*

The goal of the METRO Group Future Store Initiative[26] is to modernize the retail sector in order to increase the efficiency of warehouse management and, at the same time, create customized services and a unique shopping experience for consumers. More than 85 companies from the retailing, consumer goods, IT, and service sectors are involved in the initiative. The first Future Store opened in Rheinberg, North Rhine-Westphalia, in April 2003, and a new *real,-* Future Store has opened in spring 2008 in Toenisvorst.

The *real,-* store introduced several innovative assistance technologies for its customers. Smart scales for fresh products have an integrated camera that recognizes the vegetable or fruit customers place on the scales device. Customers no longer have to remember numbers or search the display for the corresponding product. Information terminals help customers find their way and certain products. Moreover, terminals in the music department can be used to listen to music albums or search for specific titles on the basis of their individual preferences.

In the beauty and body care department, the skincare terminal (Figure 27) provides product assistance depending on the customer's skin type, which is measured by the device through a special swap. The system suggests suitable products depending on the measurement and the client's age and sex. It is also possible to take a photo and try out eye shadows or lipsticks through the touchscreen.

Finally, customers scan their goods with a barcode scanner at self-checkout counters (Figure 28). The weight of the shopping bag is automatically compared with the scanned items. The customers are then taken through the payment steps on the display or by using an audio explanation. For mobile phones, the real,- store provides a shopping-list application that allows to simply add products to the list by scanning their barcode using the phone's digital camera.

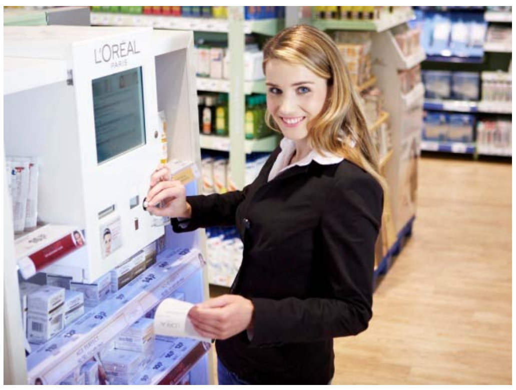


Figure 27: Skincare terminal. Image: METRO AG.

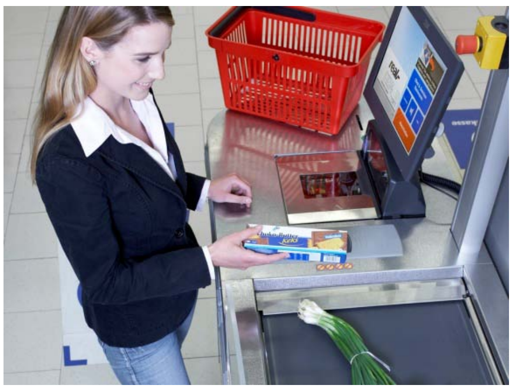


Figure 28: Self-checkout. Image: METRO AG.

[26] http://www.future-store.org/

3.1.2.2 *Innovative Retail Lab (IRL), DFKI*

The Innovative Retail Laboratory[27] (IRL) is an application-oriented living lab of the German Research Center for Artificial Intelligence (DFKI), which was installed in the head office of the German chain store GLOBUS SB-Warenhaus Holding in St. Wendel to concentrate on the demands and potentials of future retail stores. The goal of the 500 m² lab is to develop intelligent shopping consultants for customers, based on innovative interaction paradigms.

The range varies from personalized shopping assistants, "talking" products as well as intelligent shopping carts, which plan and show the way through the store according to the shopping list. Furthermore they can give advice on what to buy for certain recipes; they compare products, point out special offers in a personalized way and give additional information about the products.

From the IRL's perspective, the shopping process starts at home by checking what is needed and by adding items to the shopping list on the mobile phone, as shown in Figure 29

An intelligent sommelier terminal helps the customer to find the right wine for a meal. RFID readers detect if a bottle is taken out of the instrumented shelf, and the bottle introduces itself in an anthropomorphous style to the customer using speech. If two bottles are taken out of the shelf at the same time, a comparison chart of their features is displayed on the monitor. If a higher priced product is taken out of the shelf, general information about it is shown at first. Particle computing nodes with accelerometer sensors detect the current orientation of the bottle. If the user turns the bottle around in order to read the label on the back side, the system assumes that the user is interested in further details and proactively displays the web page of the producer. The paradigm of physical input gestures also works for other products as well, for example the user shown in Figure 30 compares cereals.

The IRL has also developed navigational assistance that helps the customer to locate products in the store. An instrumented shopping cart infers its position by reading RFID tags that are mounted to the floor and at the shelves and provides directions to the user on a display. The customer is navigated from one product to another according to the shopping list. Using a touch-screen, the customer can also select individual products from his or her shopping list. Thereupon, the customer is navigated to the product on the shortest path.

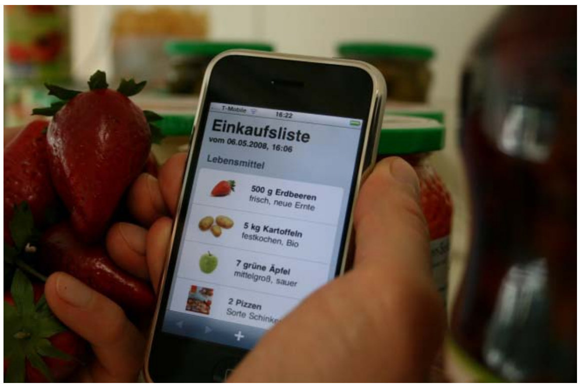


Figure 29: Shopping list on a mobile phone. Image: IRL.

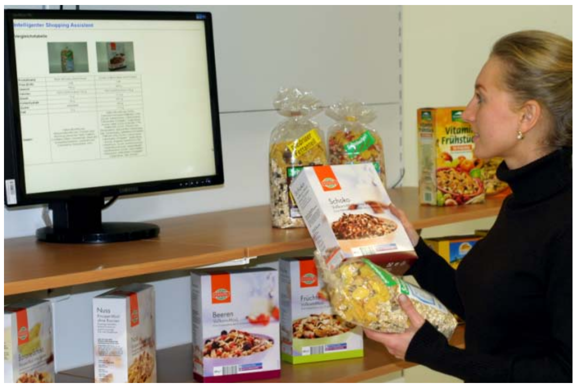


Figure 30: Interacting with physical products. Image: IRL.

[27] http://www.dfki.de/irl/uk/project.htm

3.1.2.3 *ShopLab, Philips*

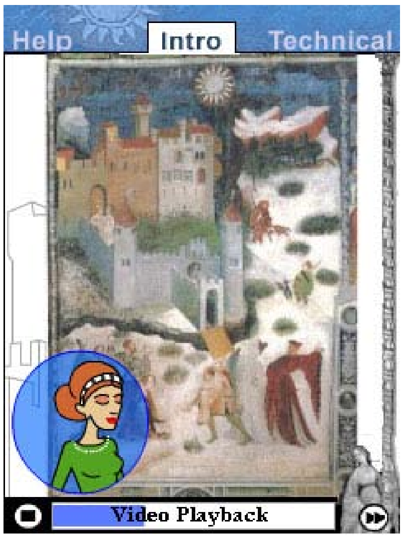
The ShopLab is also part of the Phillips Experience Lab. According to de Ruyter (2007b), the goal of the ShopLab is to enhance the shopping experience through lighting atmospheres and the introduction of interactivity through interactive shop windows, interactive signage, and reactive spots. The shop atmospheres are adapted to the people's presence and product interests while they are in or near the shop. This requires a smart environment with sensor input and actuator output. The ShopLab is used to perform user studies with retailers and end-users (shoppers).

3.1.3 Cultural Heritage

PEACH (Personalized Experiences with Active Cultural Heritage) (Kruppa et al., 2003; Stock and Zancanaro, 2007) has been a large distributed research project, lead by the Center for Scientific and Technical Research at the Trento Cultural Institute (ITC-irst). The project has been concerned with the development of an interactive and personalized tour guide that provides an educational and entertaining experience that fits the individual background and interest of its user. The PEACH project has installations at cultural heritage sites in three different countries, namely: i) Museo Castello del Buonconsiglio in Trento, Italy; ii) Völklingen Old Ironworks near Saarbrücken, Germany; and iii) Hecht museum in Haifa, Israel.

One major design goal of the PEACH guide has been to provide localized information about the exhibits that are distributed in physical space. Whereas a mobile device (PDA) is ideal to convey personalized information to its user, the small display size and limited computational power poses limitations to its use. Therefore some information is preferably displayed on large, stationary screens. The PEACH guide seamlessly integrates presentations of both mobile and stationary devices through the concept of **migrating characters** (Kruppa, 2006).

PEACH solves the localized presentation of information through infrared beacons that emit a position signal. If the user's PDA receives such a signal, the presentation on the mobile device adapts to the local, physical exhibit in front. The information is presented through virtual characters that tell stories about the exhibits and their historical background, as shown in the left screenshot of Figure 31. If the user stands in front of a stationary screen, the mobile device connects to the display and the user transfers the virtual character from the PDA to the screen. The migration of the character between the devices is indicated to the user through sound on both devices and an animation that shows how the character slowly disappears on the source device and re-appears on the target device. Stationary displays are not associated with exhibits; instead the user's character allows selecting multimedia content on the public display by pressing numbers on the mobile device, similar to a remote control. The stationary screen also hosts other virtual characters that know about different aspects of the exhibits, as shown in the right screenshot of Figure 31. The users are given the chance to exchange the virtual character that they brought with them "inside" their PDA against other characters, according to their interests.



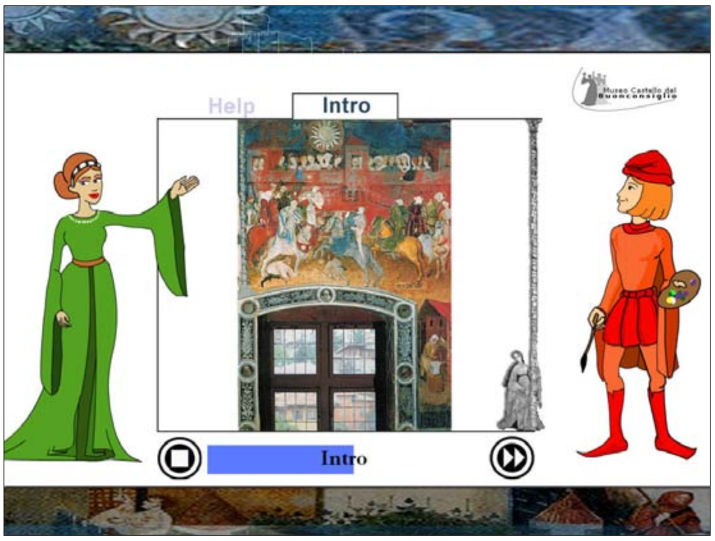


Figure 31: PEACH allows the user to migrate a virtual character from their PDA (left screenshot) to public displays (right screenshot) to control them. Each character comments the exhibits differently.

3.1.4 SmartFactory, Kaiserslautern

The non-profit registered association named "Technology Initiative SmartFactory KL"[28] was established in June 2005 by founding partners from research and industry. Their common goals were the development, application, and distribution of innovative, industrial plant technologies. Today, the challenges of computer integrated manufacturing (CIM) are the complexity of planning and operations. At the same time, the markets demand more individualized products and shorter product lifecycles. In order to demonstrate a modular factory design, a production facility has been built (see Figure 32) where bottles are individually filled with colored soap according to their internal product memory (Figure 33).

The SmartFactory aims to implement a lean production structure by means of ubiquitous computing technology, such as wireless connectivity and mobile devices. According to Zühlke (2008), lean means in this context reducing complexity, avoiding waste technologies and information and strictly supporting the humans in their daily work. Zühlke emphasizes that the factory devoid of humans is an aberration, the human should be in the center of factory operations. Future systems should focus on humans and their abilities and not conversely demand that the humans adjust to whatever technology. The general goal for the future is to improve usability of production devices by separating their UI from the hardware and to move it from fixed control panels to standardized devices such as PDAs or mobile phones. Based on location information, these mobile control devices could automatically adapt their UI to the nearby facilities. They could even deactivate parts of the production to ensure the safety of the worker, and implement access-control rules. Further benefits of in-plant positioning could be asset-management, tracking of goods and products in order to trace errors in their production, and control of automated guided vehicle systems.

Zühlke (2008) describes the driving scenario of the SmartFactory as a service task in a plant, where service staff has to replace a valve. The alarm appears on the mobile phone of the service staff, which is also able to guide the worker directly to the place of malfunction using

[28] http://www.smartfactory-kl.de/

GPS outdoors and an indoor-positioning system within the plant. Portable PC-systems will identify the broken valve and download necessary information for replacements.

In order to realize this scenario, the SmartFactory has been equipped with four different indoor-positioning systems to compare their precision and accuracy in an industrial environment that is characterized by both electromagnetic and ultrasonic noise. The first system is based on RFID tags which are integrated into the floor, so that mobile units can use antennas on the bottom to read the tags and infer their position from the IDs of the nearby tags. However, this approach implies that the location of each tag is exactly known and stored in a database. The second system is the Cricket System, which is based on the time-difference-of-arrival (TDOA) between ultrasonic signals between mobile tags and an array of fixed beacons. The third system, Ubisense, is based on ultra wide band signals that are emitted from mobile tags and received by sensors that are placed in the corners of the plant. According to Stephan et al. (2008), system performance for the UWB technology ranges from 1.30 m down to 0.12 m. For the ultrasound technology system performance ranges from 0.08 m down to 0.015 m for accuracy and precision values.

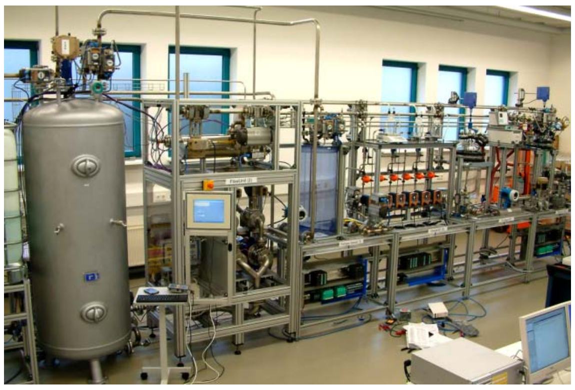


Figure 32: Demonstration facility. Image: Smart Factory.

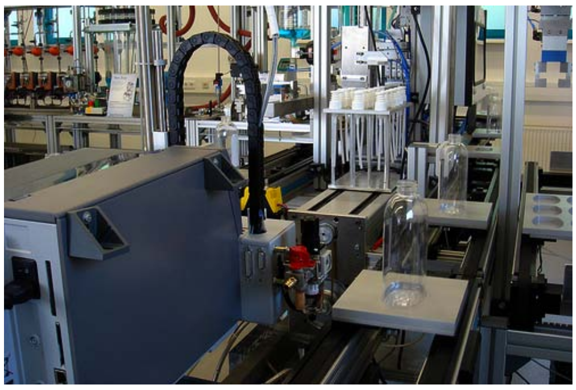


Figure 33: The process is controlled by the bottle's product memory. Image: Smart Factory.

3.2 Activity and Task Modeling

One of the basic research questions concerning this thesis is how to model and represent tasks and activities. In the following, we will present two very different approaches, one from the field of Ubiquitous Computing, and one from the Knowledge Management domain.

3.2.1 Activity Recognition with Multiple Goals

In their article, (Hao Hu et al., 2008) ask the fundamental questions of whether users often carry out multiple concurrent and interleaving activities or single activities in their daily life, and if so, whether such complex behavior can be detected accurately using sensors in an ubiquitous computing environment. They state that in order to model user's activities from observed action sequences, a deep understanding of multiple-goal pursuant behavior is essential. They distinguish between concurrent and interleaving goals; the prior means that multiple goals are pursued at the same time slice, whereas the latter means that they are pursued non-consecutively.

The authors have investigated available benchmark data from the MIT PlaceLab ("PlaceLab

Intensive Activity Test Dataset 1") that shows three hours of family activity in the lab to illustrate the taxonomic nature of multiple goals. This dataset contains a recording of activities that have been manually labeled on a low level, such as "sweeping" or "washing ingredients". Hao Hu et al. (2008) have grouped these actions into a medium level of more generalized activities like "preparing ingredients". Finally, they categorized these activities into 9 high-level activities. The resulting hierarchical taxonomy is shown in Figure 34.

The authors developed and tested an activity recognition algorithm that is able to consider multiple goals and works on WiFi location sensor information from the PlaceLab dataset. In their experiments, Hao Hu et al. (2008) achieved acceptable recognition accuracy above 80%. This highlights a strong correlation between location and activity. From their experiments, Hao Hu et al. also conclude that the accuracy of the activity recognition algorithm depends on the granularity; coarse-grained activity recognition tends to achieve higher accuracy than fine-grained ones. This seems to be a good motivation for hierarchical activity models.

The authors have arbitrarily chosen three granularity levels for activities in their taxonomy (Low-level, Medium-Level, and High-level) and don't consider any artifacts. We expect that our suggested situational semantic activity model could help to further improve the recognition results. Likewise, the principles of Activity Theory give a better rationale for the granularity levels, e.g. operations and actions.

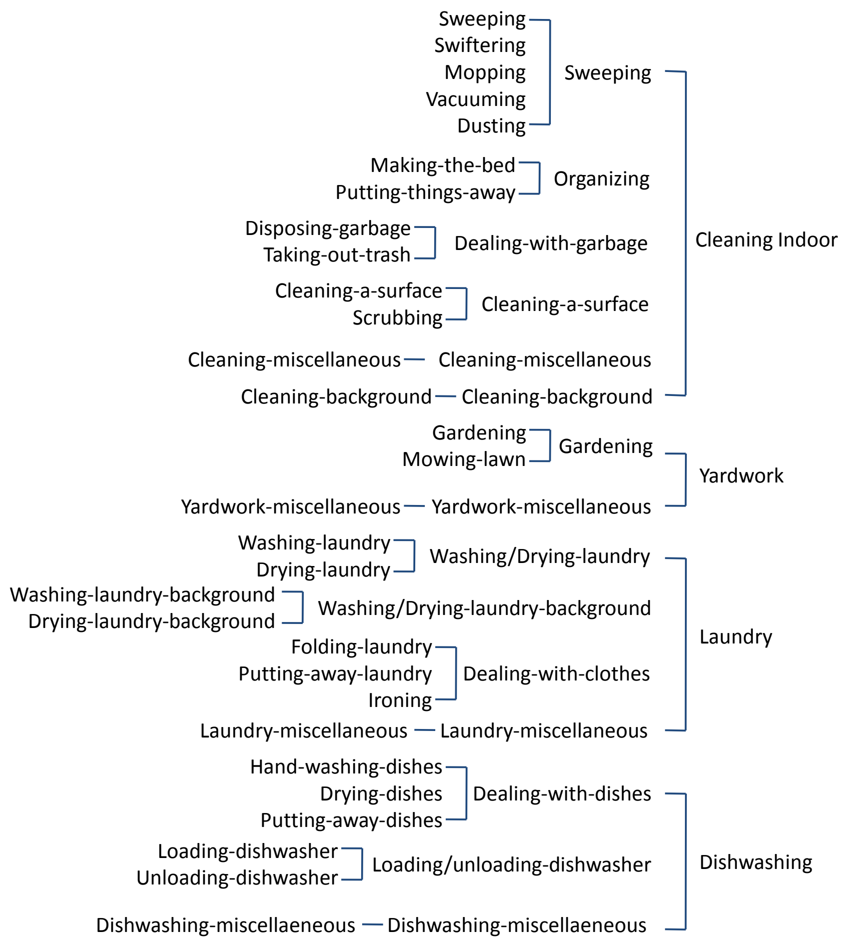


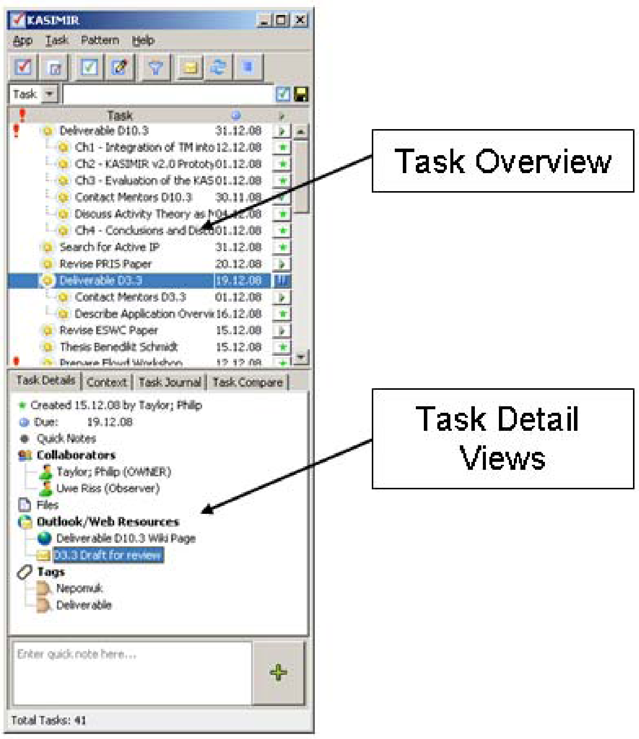
Figure 34: Taxonomy of home activity according to (Hao Hu et al., 2008).

3.2.2 Task Modeling and Management in NEPOMUK

The modeling of tasks and activities is also subject of business studies, particularly in the field of business process management. Workflow management systems (WfMS) significantly contribute to the productivity of employees. However, according to (Riss et al., 2005) their rigidity restricts their applicability for knowledge work, such as consulting and design processes, that demand flexibility, negotiation, and collaboration. On the other hand, knowledge workers often concentrate on their tasks, forgetting the organizational needs of streamlining processes. Riss et al. (2005) suggest flexible, personal task management systems that allow knowledge workers to record and share their task knowledge in the organization.

Such a task management tool, called KASIMIR[29], has been integrated in the Social Semantic Desktop (SSD) in the EU-project NEPOMUK[30] (Networked Environment for Personalized, Ontology-based Management of Unified Knowledge). The Social Semantic Desktop aims at empowering individual knowledge workers to better exploit their personal information space and to maintain fruitful communication and exchange within social networks across organizational boundaries.

KASIMIR (Figure 35) allows users to organize their to-do items and augment them with detail information from the SSD such as documents, persons, topics and others. Additional plug-ins have been developed for browsers and email clients that users to create or supplement tasks directly from the respective application.



Task	
Deliverable D10.3	31.12.08
Ch1 - Integration of TM into	12.12.08
Ch2 - KASIMIR v2.0 Prototy	01.12.08
Ch3 - Evaluation of the KAS	01.12.08
Contact Mentors D10.3	30.11.08
Discuss Activity Theory as M	04.12.08
Ch4 - Conclusions and Disc	01.12.08
Search for Active IP	31.12.08
Revise PRIS Paper	20.12.08
Deliverable D3.3	19.12.08
Contact Mentors D3.3	01.12.08
Describe Application Overvi	16.12.08
Revise ESWC Paper	15.12.08
Thesis Benedikt Schmidt	15.12.08
Prepare Flowd Workshop	12.12.08

Created 15.12.08 by Taylor; Philip
Due: 19.12.08
Quick Notes
Collaborators
Taylor; Philip (OWNER)
Uwe Riss (Observer)
Files
Outlook/Web Resources
Deliverable D10.3 Wiki Page
D3.3 Draft for review
Tags
Nepomuk
Deliverable

Total Tasks: 41

Figure 35: The Kasimir Task Management Sidebar.

[29] NEPOMUK Deliverable D3.3: Final NEPOMUK Task Management Prototype Deployed. http://nepomuk.semanticdesktop.org/xwiki/bin/download/Main1/D3%2D3/D33%2DFinalPrototypeTaskManagement%2Dv10.pdf

[30] http://nepomuk.semanticdesktop.org/

The task management model[31] of KASIMIR is based on the Task Model Ontology (TMO) that is represented by the Nepomuk Representation Language (NRL) using RDF-S. Inspired by Activity Theory, the TMO allows hierarchical task structures by the decomposition of tasks into sub-tasks. Each task refers to a goal and can have an arbitrary number of sub-tasks, and each sub-task can be delegated by its owner to one or more persons (executors). It is also possible to transfer tasks between persons. Tasks also have states; initially the task is "new", and once it has been assigned to someone, it can change between "running" and "suspended" until it is either marked as "completed" or "terminated". After acknow-ledgement through its owner, the task eventually becomes "finished" and "archived". According to the terminology of workflow systems, a completed task is also called a case. TMO also supports accurate planning by the system through an integrated time management, based on a task's duration and deadline. The system further checks whether the remaining time for a task is sufficient to accomplish all tasks in time and sends notifications, if necessary. Dependencies between the tasks can be stated through ordering: task A is precursor to B, or task A is successor to B.

As said above, it is an important requirement for the organizational task management to identify and reuse typical task patterns of workers. TMO supports the sharing of work experience on tasks with others through the Task and Pattern Repository. Pattern creation emanates from existing cases, and these patterns can be instantiated later by other users.

NEPOMUK's task model is very similar to our proposed semantic activity model in that it is based on ontological concepts and includes references to resource units (physical artifacts) and information units (virtual artifacts) in form of attachments. Whereas temporal aspects are well covered by start- and ending times (in a target and actual version), spatial aspects of tasks are not at all considered. Hence NEPOMUK is not suitable for mobile scenarios, since it cannot adapt its user interface or notifications according to the user's location.

The concept of KASIMIR equals UBIDOO, our UBIquitous to-DO Organizer, except that we explicitly model locations for each sub-task and/or relate tasks to high-level activities, so that the UBIDOO itself can suggest suitable locations nearby the user (e.g. a supermarket for basic shopping activities).

3.3 Model-based Design Methods

Generally speaking, the idea behind Model-based Design (MBD) is to design and specify complex systems on an abstract level that unburdens the designer from considering the details of the actual implementation and allows platform independent designs. MBD is used in various fields, such as software engineering, embedded systems, and interactive systems.

3.3.1 UML

The Unified Modeling Language[32] (UML) is a widely adopted industry standard that is used to

[31] NEPOMUK Deliverable D3.1: Task Management Model. http://nepomuk.semanticdesktop.org/xwiki/bin/download/Main1/D3%2D1/D3.1_v10_NEPOMUK_Task_Management_Model.pdf

[32] http://www.uml.org/

specify, visualize, and document object-oriented software systems under development. UML diagrams represent a system model from static (structural) and dynamic views. Static views include class diagrams and composite structure diagrams. They are used to define the architecture and data structures of software. Dynamic views include sequence diagrams, state machine diagrams, and activity diagrams. The latter represent procedural actions, choices, and iterations as workflows. They have no relationship to Activity Theory. UML abstracts from real programming languages and database structures and supports the design of software on a conceptual, language-and platform independent level.

3.3.2 Concur Task Tree Environment

Paternò (2000) applies Model-Based Design (MBD) to the design of interactive applications. In this context, MBD aims to find declarative models that allow designers and developers to concentrate on relevant aspects of their work without being immediately immersed in details of the implementation. The primary goal is to identify what characterizes a user interface without having to deal with platform-dependant implementation details. It is also important for the design of interactive multi-platform systems, e.g. Web portals that support mobile access. MBD is based on task models that allow designers to address the specification of functional and interaction aspects within an integrated framework. The task model reflects the needs of the users and how they interact with the system in order to achieve their goals. Hence MBD is especially suitable for user-centered design development methods.

The de-facto standard for model-based interface design is *Concur Task Trees* (CTT) that has been introduced by Paternò (1999). A CTT model hierarchically decomposes tasks into subtasks that are described by an arbitrary identifier and a category. There are four categories: i) *abstractions* are used to group tasks; ii) *applications* describe tasks that are carried out by the system; iii) *interactions* represent direct interaction between the user and the system; iv) *user* tasks describe what the user does independently of the system. Furthermore, one can specify for each task a descriptive name, how often a task is performed, a collection of required objects, and a set of preconditions. Eight relations define the temporal order in which subparts of a task are carried out, e.g. concurrent, enabling, or suspend/resume.

The *Concur Task Tree Environment* (CTTE)[33] is a graphic tool that supports the modeling of CTTs. The tool supports the creation of task models on the basis of scenarios. Paternò (2000) describes how the CTTE has been methodically applied for the design and development of an adaptive hypermedia information system for the museum of Carrara, Italy. In a first step, three different types of users (tourists, students, and experts) were involved through interviews and questionnaires to analyze the tasks that the new application should support as well as their current problems. The informal findings were then modeled using the CTT notation and the CTTE environment. The resulting task model represented the structure of the presentation and which navigation elements are required for each user group.

3.3.3 Useware Engineering with useML

We have already introduced the term useware that refers to the hardware and software components of user interfaces. In the domain of human-machine interaction, Zühlke and Thiels (2008) have introduced a structured, user-oriented *useware engineering* process that

[33] http://giove.isti.cnr.it/tools/ctte/

puts humans, their tasks, and working conditions at center stage. Similar to Paternò's work, their design method is based on a structured model of requirements and users' needs.

The first step in the suggested development process is the **analysis phase**, based on practical experience. The goal of this step the investigation of the users, their tasks, and work environment in order to identify the requirements and user's needs. An analysis tool is used to create a central development-specific database, which stores and allows access to all data accumulated during the process. The tool supports structured data management, consistent storage and evaluation of the collected requirements of the analysis phase.

The second step is the **structure design phase**. The goal is a user- and task-oriented structuring of the system usage that abstracts of any hardware or software platform dependencies. To emphasize the central importance of the user, the resulting structure is called a **use model**. The use model describes the tasks, activities, and actions of users as so called use objects, as shown in Figure 36. The use model is specified using the *USEware Markup Language* (useML) (Reuther, 2003) and can be graphically edited with the *Udit* tool (Meixner, 2009). Similar to Activity Theory, the use model is hierarchically composed. *Use objects* represent higher-level activities and actions, which are subdivided into *elementary use objects* that represent elementary actions and operations. According to Reuther (2003), the method suggests five elementary actions: change, release, select, enter, and inform. These atomic tasks can't be further divided. They represent user interaction with the system on an abstract level and lead to a consistent description of all elements. The successor standard *useML 2.0* adds cardinalities to use objects and temporal operators (choice, order independence, concurrency, and sequence). The use model describes the basic structure of the user interface (i.e. useware) and is the basis for the further development in succeeding steps of the process.

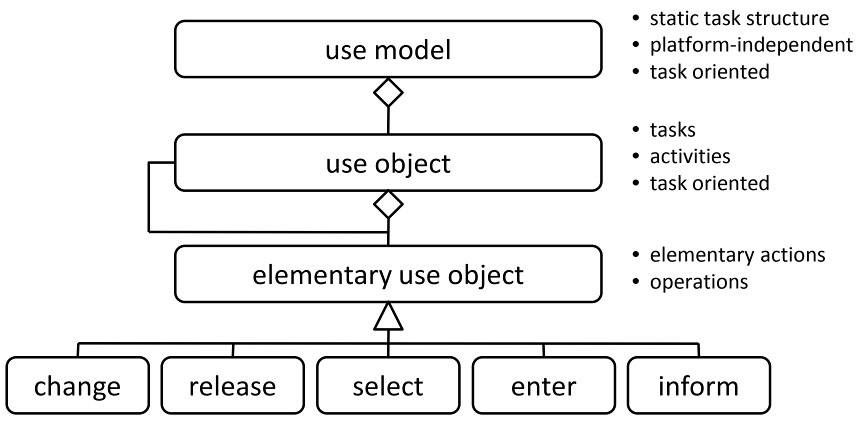


Figure 36: Use model, according to (Reuther, 2003).

The third step is the **design phase**, which is concerned with the selection of the platform for the user interface, including the computer hardware and display and interaction equipment. This step includes the allocation of display areas. Furthermore, navigation concepts and structures have to be specified for interactive systems. The elementary tasks of the use model are mapped to interaction objects (also called dialog objects). These objects can be realized through graphical dialog boxes, or other modalities like speech. Zühlke and Thiels (2008) point out that the design phase is not strictly separated from the structural design phase, and that there are strong reciprocal effects between them.

In the **realization** phase, the specified design needs to be implemented using software development environments. In many cases a media break is inevitable since these tools (e.g. Microsoft Visual Studio) do not support to import use models.

3.3.4 Discussion

The model-based development processes presented above partially cover some of our research questions. The methods presented by Paternò (1999) and Zühlke and Thiels (2008) do: i) feature a structured design method; ii) support the method through modeling tools; and iii) result in an activity-based model of the users' needs. The useML modeling language has a hierarchical structure similar to our proposed activity model. However, some important aspects of instrumented environments are not covered. Both models are focused on task-oriented, explicit human-computer interaction, whereas the focus of this thesis is on proactive, goal-directed, and implicit human-environment interaction. Also, the application domains differ. Whereas we are interested in everyday life environments, Paternò's work is focused on the design adaptive hypermedia and mobile devices, and Zühlke and Thiel are interested in the field of production environments. Furthermore, neither of the presented models covers spatial aspects of the environment. The designer has no floor plan or 3-D model at hand to support the instrumentation of the environment with hardware. The models also do not explicitly represent any situational context of tasks and activities.

3.4 Geometric Location Modeling Tools

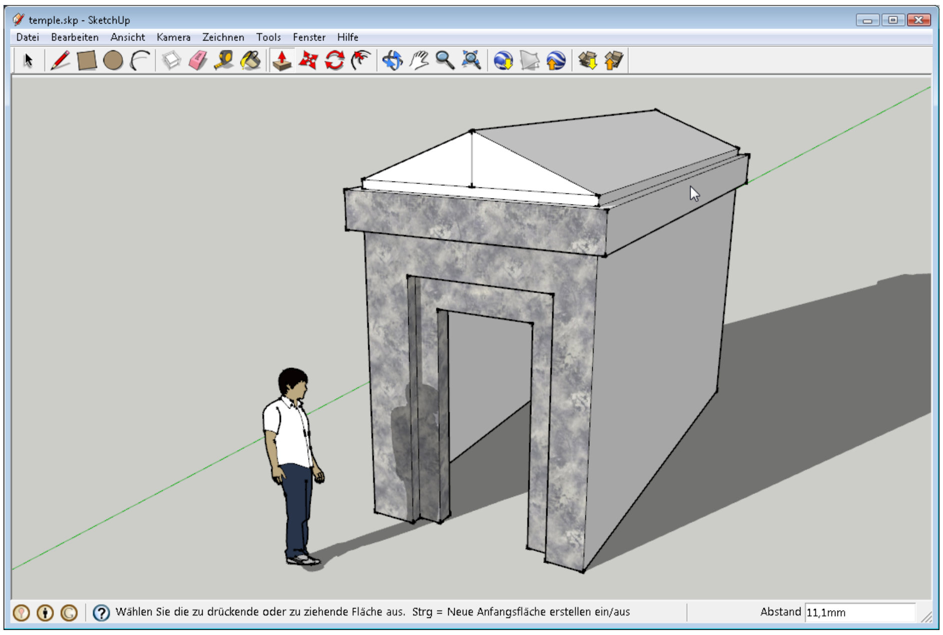
One focus of this thesis is the geometric modeling of intelligent environments with respect to their instrumentation with ubiquitous computing technology. The requirements to the location model are manifold and range from visualization and planning to indoor positioning and navigation. In the following, we will give an overview of existing modeling tools that address various aspects, from home planning support to professional drawing tools for architects. We will introduce their individual concepts and finally discuss their suitability for the modeling and simulation of ubiquitous computing environments.

3.4.1 Google's SketchUp 6

SketchUp has been developed since 2000 by the *@Last* software company as a general purpose 3-D modeling tool (Figure 37). In 2006, SketchUp has been bought by Google which made the software freely available to let people create building models for Google Earth[34]. In contrast to similar tools for professional users, such as 3DMax or Blender, SketchUp's user interface provides an intelligent and powerful set of functions that allows even novice users to create 3-D models in an intuitive way. In particular, the Push/Pull tool allows the user to sketch a shape (rectangle or circle) on any face of a solid object, and to extrude it into 3-D. By doing so, 3-D features can be easily added to objects, and also holes can be cut out. The Follow-Me tool allows one to create complex extrusions and lathed forms along predetermined paths.

SketchUp 6 provides full support for Google Earth; aerial images can be imported as a site to construct building models, and the resulting content can be published in Google Earth by using the *Keyhole Markup Language*[35] (KML). KML is an XML language focused on geographic visualization, including annotation of maps and images. SketchUp also implements an Avatar view that allows the designer to walk through building models.

The major drawback of SketchUp is the lack of any semantic knowledge about the modeled geometry. Models are a plain assemblage of edges and faces without associated meaning that do not serve any other purpose than their graphical visualization. Whilst it is simple to add details like stairs or handrails to walls using the Follow-Me function, there is no way to alter the geometry of the wall afterwards.


temple.skp - SketchUp
Datei Bearbeiten Ansicht Kamera Zeichnen Tools Fenster Hilfe

Wählen Sie die zu drückende oder zu ziehende Fläche aus. Strg = Neue Anfangsfläche erstellen ein/aus

Abstand 11,1mm

Figure 37: Modeling with Google's SketchUp tool.

[34] http://earth.google.com/
[35] http://www.opengeospatial.org/standards/kml/

3.4.2 Modeling of Complex Building Facades in Computer Graphics

Nowadays it has become a common feature of personal navigation systems to visualize buildings in 3-D as landmark objects, and also Google Earth is known for its 3-D models of certain landmarks or even complete cities, like New York or Berlin. The creation of building models however remains a tedious task with tools like SketchUp. In particular, the high level of detail of building facades (particularly classic styles of the 20th century) poses a challenge to the modeler.

Finkenzeller (2008) has implemented a system that semi-automatically applies different architectural styles to an abstract building model (see Figure 38) that can be created easily in a few steps with SketchUp. Using semantic annotation about the building, the abstract model is refined into a conceptual model that contains qualitative information about its geometry, e.g. which faces represent a door and where a balcony is located. Based on this information, the system is able to automatically create highly detailed geometry by applying rules (grammars) and pre-defined building blocks for doors, windows, and decoration elements. Figure 39 shows an example of an automatically generated building model.

Finkenzeller's work is strictly focused on facades, so he does not discuss the modeling of indoor spaces. However, it seems possible to us that his approach of a conceptual model could be transferred to create highly-detailed, representative rooms as well.

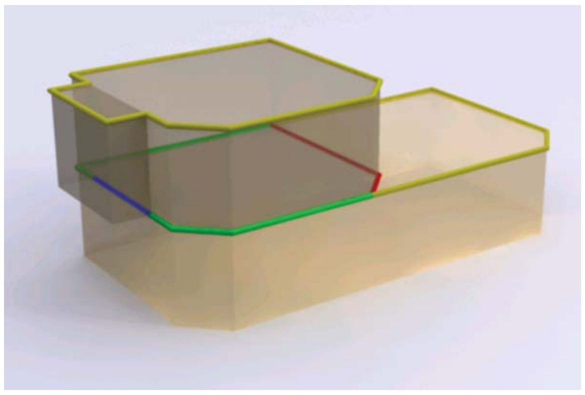


Figure 38: Abstract building model.

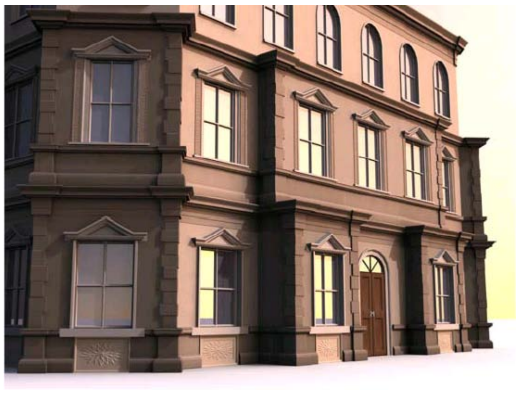


Figure 39: Generated facade details.

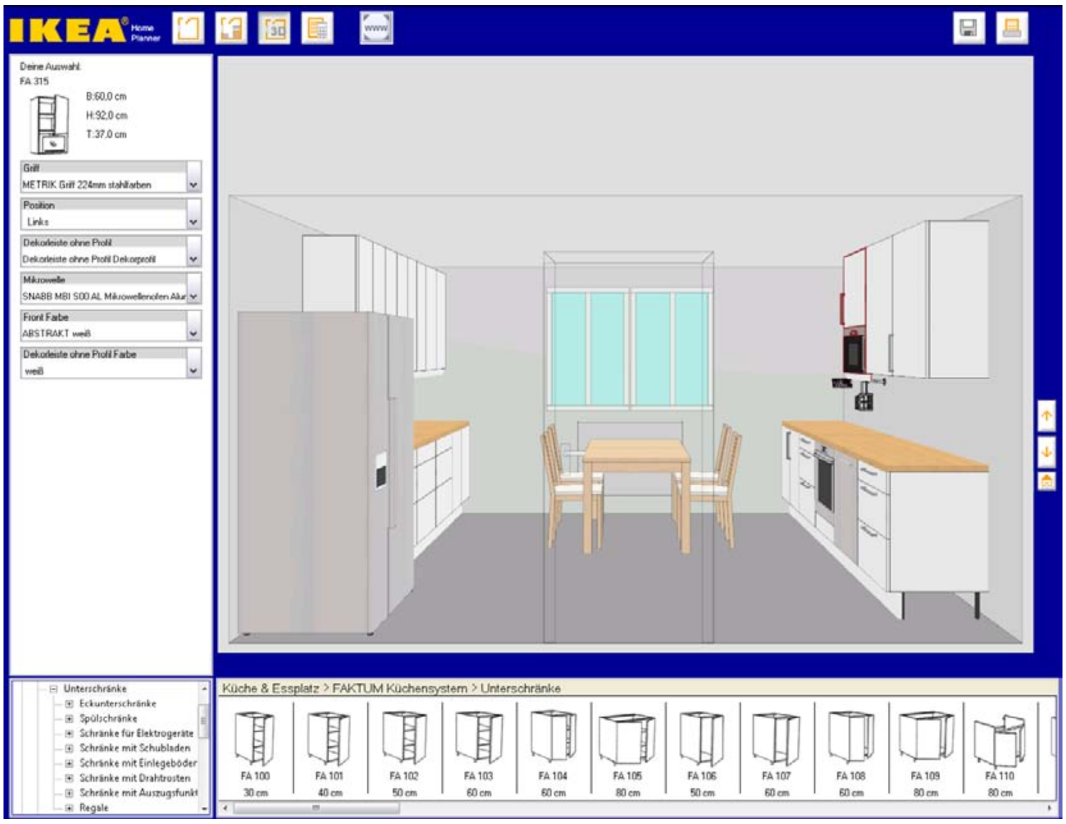
3.4.3 IKEA's Home Planning Tools

IKEA, internationally known for its flat pack furniture, offers a set of tools[36] that allows their customers to plan kitchen spaces, offices, and bedrooms based on the IKEA product range. The workflow starts with a dialogue that asks the user to enter the dimensions of the room that is to be furnished. To simplify this step, the software is limited to a set of predefined room shapes. The user can drag and drop building elements, such as doors, windows, and heating elements to the floor plan. It is also possible to define gas, water, and electricity outlets. In the next step, the user can choose from the full range of IKEA furniture and drag and drop items into the room. New items will automatically be aligned to walls and adjacent items. Cabinets can be fitted with optional storage accessories. It is also possible to try different styles by modifying the surface material and handles, considering the available

[36] http://www.ikea.com/ms/en_GB/rooms_ideas/splashplanners.html

choice. In the third step, the room model can be visualized in 3-D, where the furniture appears textured, as shown in Figure 40. If the model is completed, the actual price can be calculated to see what it will cost and the model can be uploaded to the IKEA server to buy the necessary items in a local store. The IKEA Home Planner is a stand-alone solution for the shopping process and comes without any import- or export functionality.

The planner internally clearly maintains qualitative knowledge about the furniture items, since the system considers rules regarding where accessory parts can be mounted to cabinets, and which colors/surfaces are available for which item.


Deine Auswahl:
FA 315
B:60,0 cm
H:92,0 cm
T:37,0 cm

Griff: METRIK Griff 224mm stahlfarben
Position: Links
Dekorleiste ohne Profil: Dekorleiste ohne Profil Dekorprofil
Mikrowelle: SNABB MBI 500 AL Mikrowellenofen Alu
Front Farbe: ABSTRAKT weiß
Dekorleiste ohne Profil Farbe: weiß

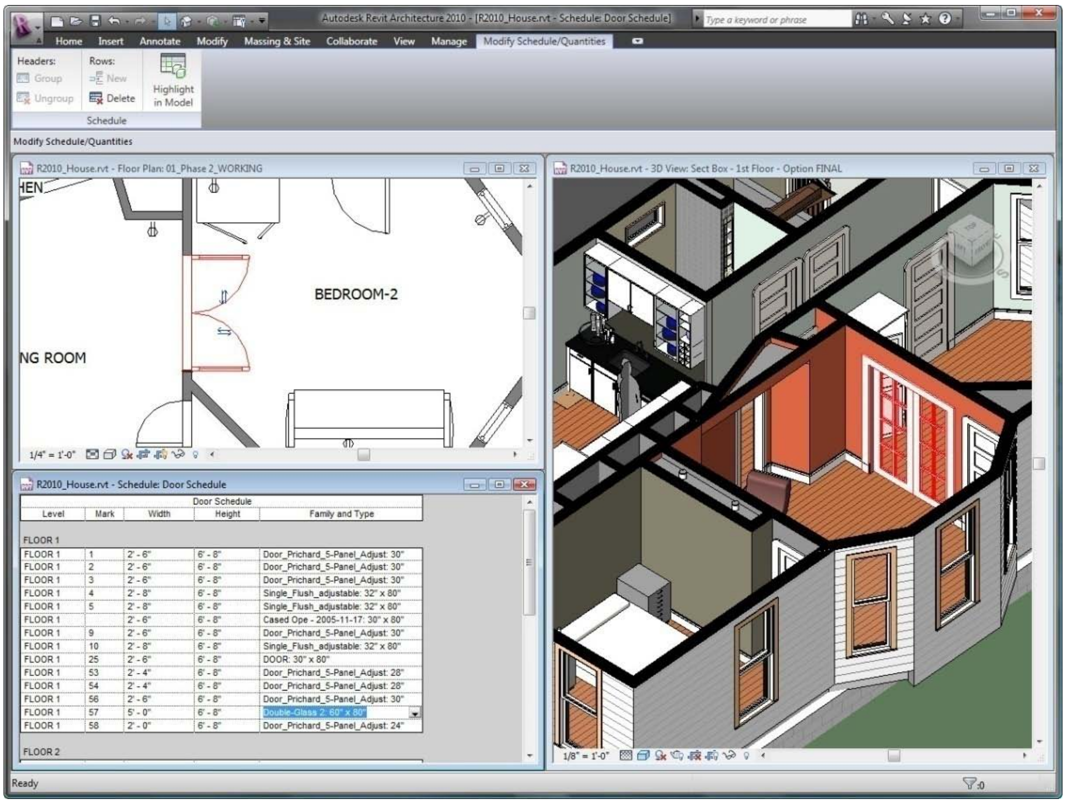
Figure 40: Editing a kitchen model with the IKEA kitchen planner.

3.4.4 Autodesk's AutoCAD Architecture and Revit

AutoCAD Architecture[37] provides several extensions to the general-purpose AutoCAD software that target the specific needs of architects. It supports the drafting of floor plans through automated elements for walls, doors, and windows. When the architect places a door or window in a wall, the wall is automatically trimmed to accept that object. The size of doors and windows can be easily changed by entering their dimensions. Because walls, doors, and windows are style-based, the appearance of any or all of them can be easily changed. AutoCAD Architect also provides elevation, section, or perspective views from 2-D designs. It is possible to export the drafts into the 3-D modeling tool 3DMax to create full 3-D building models.

[37] http://www.autodesk.de/adsk/servlet/index?siteID=403786&id=12433856

The Autodesk Revit Software[38] goes beyond the possibilities of AutoCAD and supports full 3-D modeling based on parametric objects for doors, windows, roofs, and other objects like cabinets, as shown in Figure 41. Revit follows a new Computer Aided Design (CAD) paradigm called Building Information Modeling (BIM). BIM stands for intelligent, three-dimensional, and parametric object-based design. A BIM model may contain the building's full life cycle, from concept to construction to decommissioning. Small-scale views of building components may be created using a combination of 3-D and 2-D drafting objects.


Autodesk Revit Architecture 2010 - [R2010_House.rvt - Schedule: Door Schedule]
Type a keyword or phrase
Home
Insert
Annotate
Modify
Massing & Site
Collaborate
View
Manage
Modify Schedule/Quantities
Headers:
Group
Ungroup
Rows:
New
Delete
Highlight in Model
Schedule
Modify Schedule/Quantities
R2010_House.rvt - Floor Plan: 01_Phase 2_WORKING
HEN
BEDROOM-2
NG ROOM
1/4" = 1'-0"
R2010_House.rvt - 3D View: Sect Box - 1st Floor - Option FINAL
1/8" = 1'-0"
R2010_House.rvt - Schedule: Door Schedule
Ready

Door Schedule

Level	Mark	Width	Height	Family and Type
FLOOR 1				
FLOOR 1	1	2' - 6"	6' - 8"	Door_Prichard_5-Panel_Adjust: 30"
FLOOR 1	2	2' - 6"	6' - 8"	Door_Prichard_5-Panel_Adjust: 30"
FLOOR 1	3	2' - 6"	6' - 8"	Door_Prichard_5-Panel_Adjust: 30"
FLOOR 1	4	2' - 8"	6' - 8"	Single_Flush_adjustable: 32" x 80"
FLOOR 1	5	2' - 8"	6' - 8"	Single_Flush_adjustable: 32" x 80"
FLOOR 1		2' - 6"	6' - 8"	Cased Ope - 2005-11-17: 30" x 80"
FLOOR 1	9	2' - 6"	6' - 8"	Door_Prichard_5-Panel_Adjust: 30"
FLOOR 1	10	2' - 8"	6' - 8"	Single_Flush_adjustable: 32" x 80"
FLOOR 1	25	2' - 6"	6' - 8"	DOOR: 30" x 80"
FLOOR 1	53	2' - 4"	6' - 8"	Door_Prichard_5-Panel_Adjust: 28"
FLOOR 1	54	2' - 4"	6' - 8"	Door_Prichard_5-Panel_Adjust: 28"
FLOOR 1	56	2' - 6"	6' - 8"	Door_Prichard_5-Panel_Adjust: 30"
FLOOR 1	57	5' - 0"	6' - 8"	Double-Glass 2: 60" x 80"
FLOOR 1	58	2' - 0"	6' - 8"	Door_Prichard_5-Panel_Adjust: 24"
FLOOR 2				

Figure 41: Editing a building model in Revit. Image: Autodesk.

3.4.5 LindenLab's Second Life

Second Life differs from the other tools mentioned above, since it is not only an editor, but a persistent virtual world that is accessible via the Internet through a client program. Thousands of users can visit and explore the world through avatars and interact with each other. Second Life is primarily focused on social experiences in virtual places and there are no game-like tasks or activities to engage. Every user has the possibility to buy a parcel of virtual land and can use an integrated editor to construct buildings on this land. It is also possible to create virtual artifacts, which can be traded with other users.

Objects can be interactively built from the avatar's egocentric perspective by using a palette of geometric primitives, such as box, sphere, and pyramid. These so-called prims can be modified through geometric operations like twist, taper, and shear. The world's physics engine makes it possible to animate objects with realistic physical properties and to use particles in order to simulate effects like snow or smoke for example.

[38] http://usa.autodesk.com/adsk/servlet/index?id=3781831&siteID=123112

Linden Scripting Language, also referred to as LSL, is a programming language that allows the designer to add interactive and intelligent behavior to any object in Second Life. For example, it is possible to implement light switches or doors that open when they are clicked.

By using LSL, the Fraunhofer Institut für Produktionstechnik und Automatisierung (IPA) has implemented the "factory of eMotions"[39] in Second Life, a virtual but fully-functional factory that produces car objects on behalf of the user (see Figure 42). The user can control the production process in order to build customized vehicles.

Furthermore, LSL can be used to create a link between the real and virtual world. Brandherm et al. (2008) have implemented a simulation framework to evaluate the functionality of sensor-based systems in Second Life. In order to simulate the indoor-positioning system *LORIOT*, they have modeled virtual RFID- and infrared-beacons in Second Life and fitted the avatar with a virtual PDA. Theses virtual objects send their position and orientation via LSL scripts to a so-called Twin-World Mediator that provides this information to the *LORIOT* software implementation that runs on a real-world computer. The *LORIOT* positioning algorithm estimates the position of the user based on the simulated signal-to-noise ratio of virtual beacons depending on their distance to the avatar. The resulting position coordinate is send back to the mediator and visualized in Second Life as sphere above the avatar, as shown in Figure 43. The effects of different beacon arrangements on the positioning engine's precision can be interactively tested by moving the beacons and watching the results in real time.

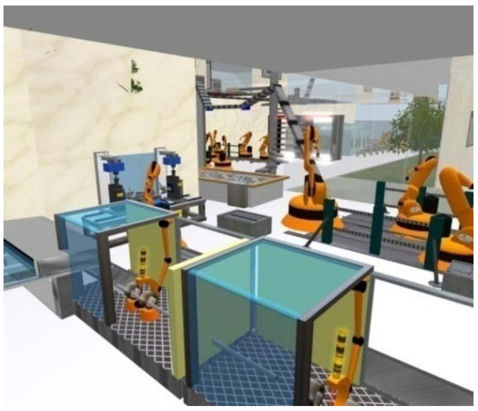
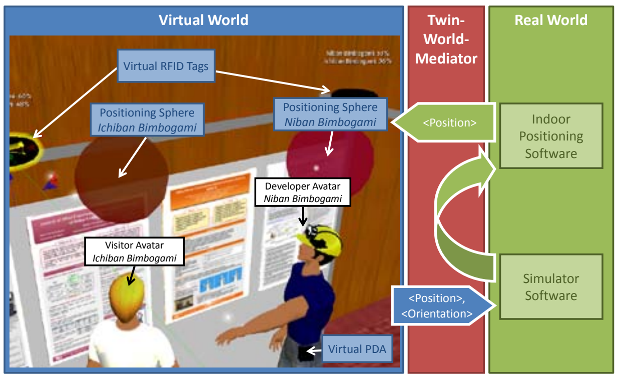


Figure 42: Factory of eMotions. Image: Fraunhofer IPA


Virtual World
Twin-World-Mediator
Real World
Virtual RFID Tags
Positioning Sphere Ichiban Bimbogami
Positioning Sphere Niban Bimbogami
<Position>
Indoor Positioning Software
Developer Avatar Niban Bimbogami
Visitor Avatar Ichiban Bimbogami
<Position>, <Orientation>
Simulator Software
Virtual PDA
Figure 43: Simulating an indoor-positioning system. Image: Brandherm et al. (2008).

3.4.6 eHomeSimulator

Armac and Retkowitz (2007) have developed the *eHomeSimulator* toolkit that allows the simulation of smart home environments. Application examples are automated lights or music that follows their listeners from room to room. The authors anticipate that it is not realistic to develop individual software for each customer; intelligent homes of the future will rather be made of reusable, standardized components. In that case, the designer's task consists of specification, configuration, and deployment of these components. Norbisrath

[39] Selber produzieren in Second-Life-Fabrik. Mediendienst der Fraunhofer-Gesellschaft, Nr. 7-2009, München.

and Mosler (2006) have introduced this process as the SCD-process, which we will now explain in more detail.

First, the designer specifies the home environment and services. During this phase, architectural information about the rooms in the house and their interconnections is captured. The *eHomeSimulator* allows the designer to represent the room layout based on a two-dimensional grid of square tiles. Each tile of the grid contains semantic annotation about the existing appliances and the passableness of that area. The resulting environment model is visualized from a top-down perspective that is externally rendered and imported from SketchUp. The rooms are overlaid by the appliances, which graphically indicate their state (e.g. light on vs. light off), and avatars that indicate the position of users. Figure 44 shows a hotel scenario of two rooms. Based on this model of the home environment, existing services specified, and new services can be added to the rooms according to the user's needs.

In the second step, the selected services need to be configured. For each location with active services, corresponding service objects are allocated. The *eHomeConfigurator* tool helps the designer to automate this task. If necessary appliances are missing, the configurator can suggest buying suitable devices. Afterwards, the services have to be parameterized, e.g. setting the preferred room temperature.

In the third phase, the already configured services are deployed in the home, initialized, and launched.

The authors have also experimented towards a Dual Reality scenario. They connected an earlier version of the eHomeSimulator, a small-scale physical model of a smart environment made of LEGO, with the Fraunhofer InHaus[40] in Duisburg. They were able to use the switches of the LEGO model to control the real lights in the InHaus environment.

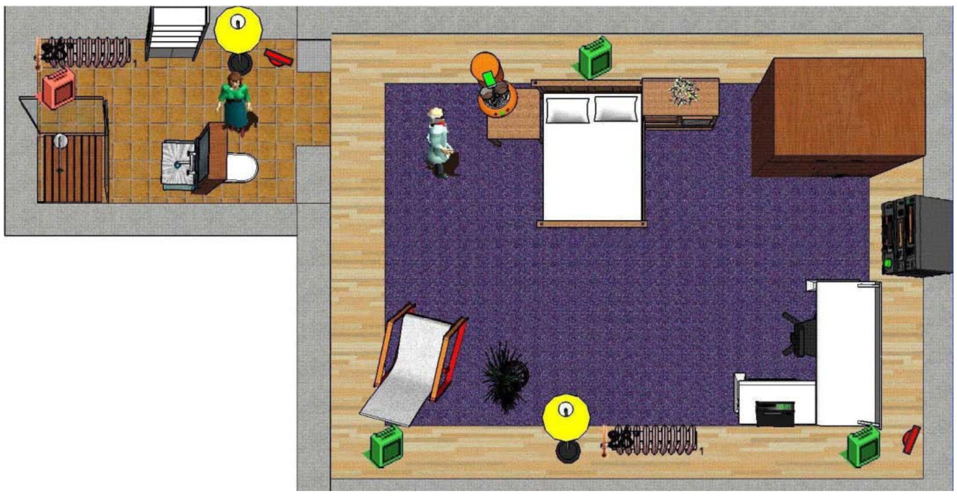


Figure 44: Model of a hotel environment with two rooms and appliances. Image: Armac and Retkowitz (2007).

[40] http://www.inhaus-zentrum.de/site_en/

3.4.7 Discussion

We have introduced six different tools for the modeling of built environments in 3-D that have specialized in various domains, from all-purpose software to the simulation of interaction in instrumented environments. In the following, we will compare them to our own *YAMAMOTO* modeling toolkit (see Chapter 7) with respect to the modeling concepts, visualization techniques, semantic knowledge, and simulation capabilities. We will also discuss their applicability to Dual Reality scenarios, as summarized in Table 2.

Considering the modeling capabilities, SketchUp is the most versatile tool that can be used to model almost anything in 3-D. The Facades tool is not really a modeling tool, but rather specialized in the automated creation of complex geometry. IKEA's Kitchen planner and Autodesk's Revit software are professional tools to create precise and realistic architectural models. The prior is targeted to consumers and has a very limited functionality, whereas Revit is intended for professional architects and comes with a highly complex user interface. SecondLife (SL) is primarily a game engine with integrated modeling tools that allows the users to build their own structures in the virtual world. The eHomeSimulator is, like our own *YAMAMOTO* toolkit, targeted towards the modeling and simulation of instrumented environments and addresses the research community. The eHomeSimulator has its focus on simulation, so it uses only a rudimentary 2-D model, whereas *YAMAMOTO* has been designed to model multi-level environments and their furnishing in 3-D.

The presented tools also differ in their visualization style. SketchUp features a sketch renderer with shadows, whereas the other tools use default techniques, like flat shading and image textures. Facades, Revit, and *YAMAMOTO* additionally provide a non-interactive, high-quality visualization that renders reflections and smooth shadows through raytracing.

Whereas the all-purpose tool SketchUp has no semantic knowledge about the modeled geometries, the specialized tools' strength is grounded in their capability to describe the modeled scene through concepts on a qualitative level. In SL, all objects can be animated though the Linden Scripting Language; it is up to the programmer to implement the objects' semantics. Similarly, the eHomeSimulator is based on virtual device drivers that mimic the behavior of real sensors and actuators in a smart home network. The Kitchen Planner implements the IKEA product range as classes, which can be instantiated and configured by the user. The planner constraints the selection according to symbolic knowledge about the actual IKEA product catalog. In Revit, the building structure is represented as a 2-D draft that is made of parametric building elements, such as walls, windows, and doors. These elements can be rendered in different styles and customized according to parameters like thickness, width, depth, and height. From this information, 3-D geometry is created and visualized. *YAMAMOTO* follows the same approach. As a unique feature, *YAMAMOTO* also supports route finding through the annotation of the building's passableness and links all geometric objects to symbolic counterparts that are represented in an ontology.

Concerning Simulation capabilities, *YAMAMOTO* implements an interactive avatar that can be controlled by the user. Such an avatar is also part of SketchUp and SL. In addition, SL offers a simplistic physical model. *YAMAMOTO* models can be exported and explored in the dynamic kinematic SimRobot simulator. Similar to the eHomeSimulator, is possible to simulate the behavior of sensors and actuators.

Due to the sensor and actuator modeling, both systems, the eHomeSimulator and

YAMAMOTO, lead towards a Dual Reality scenario where the virtual representation of the environment is in sync with reality. In *YAMAMOTO* this is partially achieved with virtual proximity sensors that control public displays in the same way as the real sensors do. Vice versa, simulated displays in the virtual environment show the same content as the real displays do. The eHomeSimulator would be able to control real environments by replacing the driver code for the sensors and actuators, since they abstract from the protocol and device level. It would also be possible that the simulator responds to real sensors.

Table 2: Comparison between six modeling tools and YAMAMOTO.

	SketchUp	***Facades***	***IKEA***	***Revit***	***SecondLife***	***eHomeSim***	***Yamamoto***
Geometric Modeling							
Dimensions	3D	3D	3D	3D	3D	2D	2.5D / 3D
Domain	General purpose	Building Facades, Roofs	Kitchen, Bedroom, Office	Building structure, Interior design	Indoor, Outdoor, Artifacts	Intelligent Environm.	Indoor, Outdoor, Artifacts
Target User	Amateur	Pro.	Consumer	Pro.	Consumer	Research	Research
Multiple levels	Layers	+	-	Layers	+		+
Import	-	SketchUp	-	DWG, 3DS	(+)[3]		RoSiML
Export	Google Earth	RenderMan	-	DWG, 3DS	-		RoSiML, VRML
Advanced Visualization							
Shadows	+	(++)[1]	-	++	-	-	(+)[2]
Reflections	-	(+)[1]	-	-	-	-	(+)[2]
Semantic Knowledge							
Concepts	-	Faccade structure, styles	Products, surfaces, configu-rations	Building structure (room, wall, door, window), furniture	(+)[3]	Switches, Sensors, Actuators	Building structure, furniture, displays, sensors, actuators
Parametric Objects	-	+	+(surfaces)	+	-	-	+
Passable-ness / Route finding	-/-	-/-	-/-	-/-	-/-	-/-	+/+
Ontology	-	-	-	-	-	-	+[4]
Simulation							
Avatar	Walk	-	-	-	Walk, Fly, Edit	Walk	Walk, Edit
Physics	-	-	-	-	+	-	+(SimRobot Engine)
Sensors/ Actors	-/-	-/-	-/-	-/-	(+)[3]/(+)[3]	+/+	+[6]/+[7]
Dual Reality							
Real→Virtual	GPS trace in Google Earth	-	-	-	Diverse[3]	-	Position, Display,
Virtual→Real	-	-	Ordering at the store	-	(-)[3]	Controlling real lights by model swtich	Proximity Sensor

[1] external rendering with *RenderMan*; [2] external rendering with *Persistance of Vision*;

[3] possible with *Linden Scripting Language*;

[4] UbisWorld Ontology; [5] *PathFinder* component; [6] Bluetooth proximity sensor, camera, GPS; [7] beacons, displays

4 User Assistance

Figure 45: Clippy, the former Microsoft Office software assistant.

Unfortunately, Clippy lacked the elaborated decision-theoretic methods of the LUMIERE research project at Microsoft Research (Horvitz et al., 1998) and interrupted the user too often with inadequate questions. People felt bothered, so Clippy has been removed from the Office suite.

In this chapter we will introduce user assistance systems and explain how their usability can be improved through user modeling. We also discuss the advantages of goal-based and proactive interfaces. Finally, we will introduce the new concept of activity-adaptive systems.

User assistance is a general term that spans a large domain from help systems for desktop applications over location-based services that assist their users in achieving everyday goals to instrumented environments that are able to sense their users' needs. The most basic type of assistance is the printed manual that describes how to operate a product. As desktop computers became more powerful in the past, the functionality of the applications grew rapidly and the printed documentation soon exceeded hundreds of pages, so that software developers started to include the documentation in the application itself. Microsoft introduced such a help system that is based on hypertext documents and allows for a keyword search. The problem with such systems is that the user must already have special knowledge to ask for the right terms or commands. In the case of UNIX, Wilensky et al. (1984) refer to the problem of testing if two files are equal. Since the user does not know the name of the appropriate *diff* command, retrieving online help by command name would not be applicable. The *Unix Consultant* (UC) addressed this issue by a goal-based dialogue in natural language that supports questions like *"How can I find out if two files are the same?"* (Wilensky et al., 1984). Whereas the UC has been an essentially passive system built for question answering, the *Sinix Consultant* (SC) by (Wahlster et al., 1988) implemented an additional active mode where the SC observes what the user does and occasionally suggests a better way of achieving the same goal.

With the advent of graphical user interfaces, Microsoft added anthropomorphous agents like the infamous Clippy character (Figure 45) to their office software products. Clippy is an animated paper clip who tries to proactively help the user in finding the right functions in the *Office 97* suite as soon as he assumes that the user has trouble in achieving their goals.

Nowadays, mobile and ubiquitous computing raises new opportunities for help systems. Mobile computing devices are available everywhere, and can be applied to assist their user in everyday tasks besides the desktop. In cars, embedded computers of the first generation inform us about the average fuel consumption and tell us how far we can go with the remaining fuel. Navigation systems which guide us to our destination are the next step in user assistance. Beyond navigation, the latest generation of luxury sedans is fitted with pre-collision systems which do not only sense objects in the environment around the car, but also track the driver's viewing direction in order to recognize dangerous situations. Typically, a front-mounted millimeter-wave radar sensor constantly monitors the distance and closing speed of a vehicle ahead. When the system's computer determines that a frontal collision is unavoidable, the pre-collision system preemptively tightens the front seatbelts, prepares the brake assist for increased braking pressure and also tightens the wheel suspension to prepare for an evasion maneuver.

In the near future, cars will also use wireless communication technologies to share their sensor data about road and traffic conditions and warn each other in case of an emergency situation. Besides safety, assistance systems will soon be able to unburden the driver from routine tasks as steering and parking. Some cars already provide a lane driving assistant which visually tracks the markings on the road and controls the steering wheel to keep the car on its track. Other cars are able to automatically park in a narrow spot. It is an ongoing discussion whether the driver should be given the opportunity to hand over the control over the car to an autopilot-like assistant; this is a challenge for jurisdiction. Many drivers say that they would feel disempowered by such systems and probably would reject these technologies. From a technical point of view however, the systems are already feasible under ideal road conditions.

Whereas cars already include an impressive amount of sensors and proactive assistance systems, smart home environments are still in their infancy. Today, home automation is mostly used in industrial buildings for energy efficiency, e.g. to automatically turn off all lights in an office after work, and in luxury apartments due to the high price of the necessary components for such an installation. In the near future, home automation will gain importance in a third sector, namely assisted living environments for seniors. The goal of assisted living projects, such as the BMBF's German Ambient Assisted Living (AAL) Joint Programme[41], is to instrument homes with assistance systems that support the well being and independent living of elderly people in their home environment. If actuators are combined with intelligent user interfaces, home automation can allow physically impaired users to control their home environment; e.g. to open windows blinds or to configure their bed through gesture and speech, even from a wheelchair. Similarly, cognitive assistance systems can support people with mild dementia in their activities of everyday life, such as cooking.

Intelligent environments are though not only of interest for the elderly. Younger people might also benefit from additional comfort through intelligent home environments. Automatic lights and intelligent climate control systems for efficient heating and cooling can contribute to save ecologic resources, and, considering the increasing energy price, soon become a reasonable investment. Besides, the average user might also desire intelligent home entertainment systems where technology disappears and devices automatically share

[41] http://www.aal-deutschland.de/

media and configure their input and output channels according to the user's situation.

However, as we envision new assistance systems with lots of interesting functions, we should be aware of the problems that their user interfaces might pose to their users. This is especially true in AAL environments. For example, an installation of multiple digital thermistors in the home can lower the energy consumption, but programming day and night schedules can be tricky for the user. In the worst case, even simple actions like adjusting the room temperature could require the user to consult the manual. The user may experience a high effort to learn about using the new system; in the worst case, the user abandons it. Of course, this is mostly because the user interfaces of industrial products today are limited to a few buttons and small displays. By employing connectivity through the internet and open standards, richer user interfaces can be realized on a PC, like the Siemens *@home* series of household appliances. Such advanced user interfaces can guide the user through the necessary settings in a goal-directed dialogue. Several research projects like *SmartKom* (Wahlster et al., 2001) or *EMBASSI* (Herfet et al., 2001) have shown how natural language and speech technologies can be integrated with a graphical user interface to support multiple modalities for interaction. If combined with knowledge about the users and their tasks, systems can automatically adapt to the users' preferences and their situation by choosing the best settings. Ideally, a smart system with access to sensors in the environment and fitted with artificial intelligence would automatically recognize the needs and intentions of the user. For example, vision-based sensors can be used to recognize the user's mood from the facial expression, and everyday objects can be augmented with accelerometer sensors to recognize certain motion gestures. If such technologies are combined, ensembles of multiple sensors allow intelligent environments to analyze the user's actions for typical activity patterns and infer potential goals. Based on this knowledge of the user's behavior and the state of the environment, the system is able to foresee when user assistance is needed, and the system can proactively offer assistance in the sense of an agent. This is convenient for the user, who does not necessarily have to know how to explicitly interact with the system or how to use certain system functions. In the best case, the user will always receive the information that is required without having to ask for it.

In the following section (4.1) we will show how the usability of user interfaces can be improved through user modeling, learning, and adaptation to individual users and their situational context. In the second section (4.2) we will introduce goal-oriented interfaces which allow users to directly formulate their goals and automatically perform the necessary actions on behalf of the user. Finally, we will have a look at interface agents which proactively provide information to the user based on the recognized situation and intention.

4.1 Adaptation to the User and Situational Context

The people who use computers and assistance systems are individuals, they differ in their level of knowledge, personal preferences, cultural background, and attributes, like age or gender. These differences make it difficult for a developer of a system to design the user interface, content, and functionality so that it fits all possible requirements. For example, elderly people require larger font sizes so that they can read the display without wearing glasses. Children would expect more playful presentations in a museum than adults, and people who are acting under time pressure prefer a more efficient interface than others who want to spend their leisure time. Mobile devices, such as mobile phones or PDAs, can be used anywhere in completely different situations, which even increases the complexity of their design requirements. If we compare mobile phones with cord-free phones at home, we can easily make out many important features that have been introduced to adapt them to different situations of use. For example, we can lock the keyboard so that we do not call people by accident if the phone is in our pocket. Almost every mobile phone also provides a set of profiles which adapt the ringtone and vibration alarm to situations like meetings, where noise would be disturbing. Finally, the network provider, signal strength, and battery level are so vital for mobile devices that they are always indicated on the main display. We have to manually adjust all these settings many times a day; if we don't do so, we have to suffer from embarrassing moments: in meetings if the phone has not been muted, missed calls if the phone is still muted afterwards, and random calls made out of our pockets. Unfortunately, many people even don't know about the possibilities to adjust their phones to their needs because they refuse to work themselves through extensive manuals.

Maes (1994) has very early criticized function-oriented user interfaces which are based on the prevailing paradigm of direct manipulation, since they are too difficult to handle for untrained users; they don't know about the possible actions and how to perform them. Today, this argument is even stronger, since more and more untrained people are suddenly confronted with user interfaces in their phones, cars, and home entertainment systems, and unable to control them.

In the following section (4.1.1) we will give an introduction to systems which automatically adapt themselves to a model of their individual user and changing situational context. We explain how agents can be employed in the user interface that learn the behavior of the user and automatically suggest actions in order to assist in routine tasks. In the second section (4.1.2) we discuss context aware systems which are able to sense their context, most importantly location, and adapt themselves to different situations of usage.

4.1.1 Adaptation to the Individual User

From the experience with early natural language-based assistance systems, like the aforementioned Unix Consultant (UC), it became apparent that such an interface must implement some extra-linguistic mechanisms to infer that the user possesses some goals other than the one stated in the request. If the system lacks common-sense knowledge about background goals, it might suggest actions that conflict with the actual interests of the user. Wilensky et al. (1984) explain this problem by the example of a user who asks for more disk space. The system could recommend a command that deletes all files to achieve this goal, but this is certainly not what the user would consider as beneficial.

In conclusion, Wahlster and Kobsa stated that

> *"A cooperative system must certainly take into account the user's goals and plans, his/her prior knowledge about a domain, as well as false conceptions a user may possibly have concerning the domain. Thus it is no longer only the user's task to construct a mental model of the technical functioning of the system. Instead, it should also be up to the system to form assumptions about what the user believes, wants, and plans, i. e. to develop a model of the user." (Wahlster and Kobsa, 1989:5)*

Wahlster and Kobsa (1989:6) further give the following definitions:

- A **user model** is a knowledge source in a natural-language dialog system which contains explicit assumptions on all aspects of the user that may be relevant to the dialog behavior of the system. These assumptions must be separable by the system from the rest of the system's knowledge.
- A **user modeling component** is that part of a dialog system whose function is to incrementally construct a user model; to store, update and delete entries; to maintain the consistency of the model; and to supply other components of the system with assumptions about the user.

These definitions conceive a user model (UM) as a model that an intelligent program constructs of the person with which it is interacting.

Whereas the user model originally has been introduced as an additive to avoid side-effects of inappropriate system behavior, Jameson spotlights user modeling and user adaptivity in his definition of user-adaptive systems (Jameson, 2006:2):

- A **User-Adaptive System** is an interactive system that adapts its behavior to individual users on the basis of processes of user model acquisition and application that involve some form of learning, inference, or decision making.

The goal of a User-Adaptive System (UAS) is to learn something about each individual user and adapt its behavior to them in some nontrivial way. Depending on their function and form, systems that adapt to their users have been given labels ranging from *adaptive interfaces* through *user modeling systems* to *software agents* or *intelligent agents*. Whereas it is the ambitious goal for software agents to predict and decide fully autonomously which actions to take in a new situation, we will begin here with the most general concept of User-Adaptive Systems and see how adaption can improve user interfaces in different ways.

The Figure 46 below shows the general schema for the processing in a User-Adaptive System. In the process of user model acquisition, the system $\mathcal{S}$ performs some type of learning and/or inference on the basis of the information about the current individual user $\mathcal{U}$ in order to arrive at some sort of user model, which in general concerns only limited aspects of the user that are relevant for the application. In the process of user model application, the system applies the user model to the relevant features of the current situation. As outcome, the system makes some predictions or decisions about the user in order to determine how to adapt its behavior to the user.

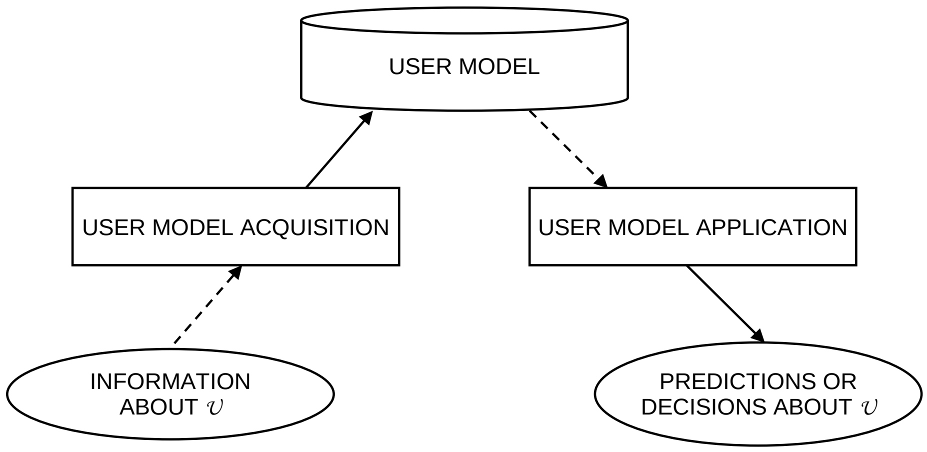

USER MODEL
USER MODEL ACQUISITION
USER MODEL APPLICATION
INFORMATION ABOUT $\mathcal{U}$
PREDICTIONS OR DECISIONS ABOUT $\mathcal{U}$

Figure 46: General schema for the processing in a User-Adaptive System. (Jameson, 2006:3)

This definition distinguishes UASs from **adaptable systems** which the individual user can explicitly tailor to their own preferences by manually choosing options that determine the appearance and behavior of the system. A strong point for adaptability is that it leaves the user in control which is a major usability concern. But although adaptability is often an attractive alternative to adaptive systems, it has several typical limitations. The user of a system may not know what options exist, since users rarely read completely through the manual. Even if the user knows or assumes that an option exists in the system, the user may not know where to find the settings in the menu structure and may also have no idea what is the best setting. According to recent usability studies, typical mobile phone buyers are frustrated if it takes more than 20 minutes to adapt the mobile device to their needs and they either don't make use of all features, or return the device. A good compromise between adaptable and adaptive systems is to let the system automatically detect possible improvements and suggest the user to apply the changes for them. Now the user has the final decision to apply the suggested changes, or rather to keep everything set as it is.

The usual way to obtain information about users is through explicit self-reports and assessments, which require the user to fill out forms with address or profession information, etc. Furthermore, any natural occurring interaction with the system can be interpreted as implicit input, for example the browsing and navigation behavior. If the user is following links in a hypertext document, the system can infer a general interest in the topic. Even subtle operations, like scrolling down a page, can indicate that the user is reading the page. The use of implicit interaction will be discussed in more detail in Section 4.2.3.

4.1.2 Adaptation to Emotional States and Cognitive Resources

Modern technologies allow the designer of User-Adaptive Systems to leave the scope of the desktop user interface and utilize various sensors to continuously monitor the psychological state of the user. Input from video cameras can be analyzed for facial expressions, and microphone data can be analyzed for certain voice features, like pitch and intensity. These findings can be supplemented by physiological data from bio-sensors, such as electromyogram signals, galvanic skin response, blood volume pressure, and pattern of respiration. Recent research has been successful to recognize emotions, like anger, sadness, frustration, joy, and adoration. Another very interesting variable is the stress level, i.e. caused by time pressure, which has a strong impact on the user's available processing capacity. If the user has a high cognitive load, the system can adapt itself and ease the situation of the user by reducing all unnecessary interaction. The related concept of resource-adaptive systems will be introduced in Section 4.1.5.

User-Adaptive Systems can improve human-machine interaction in many ways, which can be split in two groups according to Jameson (2006): i) supporting system use and ii) supporting information acquisition.

The first group subsumes system functions, in which user adaptivity can be helpful to support a user's efforts to operate a system effectively and successfully. These are:

- **Helping the user with routine tasks.** Systems assist their users in daily routine tasks through learning the users' individual habits and preferences. For example, the email-agent *Maxims* (Lashkrai et al., 1994) learns how to deal with incoming mails based on previously observed user actions, see Section 4.2.3. Today, junk mail filter functions have become an integral part of most popular email clients, such as Mozilla

Thunderbird[42]. The filter tries to automatically recognize and remove spam mail from the inbox based on the similarity to previously deleted messages.

- **Adapting the user interface**. Familiar examples are smart menus which initially hide most of the options which are rarely used by untrained users in order to reduce the complexity of the menu trees. Advanced users have the chance to expand the menus in order to access all available options. As the user begins to use hidden options, the menu structure will change and the once hidden options will now be permanently shown. Likewise, the list of options can be sorted by the frequency of use. Especially for mobile devices with small displays, this technique can effectively reduce the number of steps needed to scroll through a list of options. On the downside, frequent changes of the menus lead to the usability problem of the system's predictability.
- **Giving advice about system use.** The complexity of technology is steadily increasing as our demands grow, so that even formerly simple objects, like television sets become home entertainment systems that require experts to set them up. So, nowadays the user not only needs advice how to use desktop computer applications, but also any electronic device. The *Lumiere* help system (Horvitz et al., 1998) has been designed to spontaneously present suggestions to the user, as it observes behavior that indicates that the user might require help. This type of adaptive help is especially useful if the user is not familiar with the concepts and terminology, so that the user is not able to make use of the documentation by looking up a topic or through a directed keyword search. However, spontaneous intervention raises usability problems of unobtrusiveness and controllability.
- **Controlling natural language dialogs**. Natural language interfaces have great potential for untrained users to lower the barrier of interacting with computers, especially in conjunction with input modalities like speech, mimics, and intra- and extra-gestures. However, the analysis of these input modalities is typically limited by uncertainty and ambiguity. Particularly in case of unexpected noise the recognition rate of an (untrained) speech recognizer drops. In such a case, a natural conversation with rich vocabulary and grammar is not possible. The problem can be mitigated if the system responds to low confidence values by adapting its dialog strategy to a much simpler mode of conversation, where the user input is restricted to simple “yes” or “no” replies. A reduced vocabulary can also help to avoid ambiguity. Speaker Classifaction (Müller, 2007) techniques can be applied exploit the so called para-linguistic information of a speech about the speaker's anatomy, physiology, linguistic experience, and mental state. These characteristics are found in speech at all levels, from the spectral information in the sounds to the choice of words and utterances themselves. The *Agender* system (Müller, 2005) for example is able to estimate the speakers' age and gender of a speaker. The target applications of *Agender* involve mobile shopping as well as pedestrian navigation systems. On the basis of the user models that are provided by *Agender*, the shopping assistant is able to make a specific selection of gender-specific and age-related products (e.g. digital cameras) or services. For example, when the speaker is recognized as being a female person, the system can choose a camera that is especially designed for women. Another example

[42] http://www.mozillamessaging.com/en-US/ visited 10.07.2009

is that a tourist guide system can adapt the selection of alternative routes: when the user is a child, the guide can choose sights that are especially interesting for kids.

The second group relates to the task of information acquisition, where information overload is a major problem. Adaptation to the user model can help people to find what they need:

- **Help the user to find information.** A system that learns and knows about the interests of an individual user is able to guide the user towards interesting bits of information. The system can support both browsing and query-based searching activities, as well as it can try to proactively provide interesting information to the user (see also Section 4.2.3).
- **Tailor information presentation**. Search engines or e-commerce systems often return many pages full of results, which do not entirely fit on the screen. Especially on mobile devices, their small displays are a limiting factor for browsing interaction. User-Adaptive Systems can decide which information is relevant to the user and hide unnecessary details to save screen space.
- **Recommending products or other objects.** Online shoppers who are not familiar with the concepts used to characterize products (e.g. technical terms) may have difficulties to make an informed buying decision. They might also be simply overwhelmed by the available choice. The common strategy to ease these problems is to collaboratively recommend and filter for products called *neighbors*, which have been liked or bought by other users with a similar taste. From the vendors' viewpoint, it is an obvious benefit that users find more products they consider worth buying.

4.1.3 Usability Challenges

Adaptive interfaces have to be carefully implemented, since they might interfere with the users' assumptions about traditional computer interfaces and can lead to misunderstandings and frustration if they do not work as expected. We have seen in the introduction of this Chapter from the example of Microsoft's poor implementation of the *Office Assistant* that the users responded very negatively to inappropriate interruptions from *Clippy* and turned the assistance system off. To avoid such failures, Jameson (2006) summarizes five major usability challenges that are especially important to adaptive system functions:

- **Predictability and Transparency**. The user should be able to predict the outcome of their actions and have a clear mental image about how the system works. The layout of interface elements should be fixed, so that experienced users can internalize frequent actions. In terms of activity theory, these actions become operations which can be performed unconsciously. It should also be possible for the user to estimate the overall competence of the system, so that it will not be over- or underestimated.
- **Controllability**. Controllability refers to the extend to which the user can bring about or prevent particular actions or states of the system. A natural way to put the user in control is to ask the user for confirmation of each adaptation. However, this solution conflicts with the goal of unobtrusiveness.
- **Unobtrusiveness**. Obtrusiveness refers to the extend to which the system places demands on the user's attention which reduce the user's ability to concentrate on the primary task. If the system prompts the user too often, the additional actions

may outweigh the benefit of the adaptation.

- **Privacy**. Especially in e-commerce applications, personal data about the user is collected by people who are likely to use the data in ways that are not in the interest of the user, such as marketing. There is no straightforward solution to this dilemma.
- **Breadth of experience**. By delegating tasks to the system, the users might miss the chance to learn about the domain by browsing through the possible options by themselves. Systems can mitigate this problem by intentionally proposing solutions that are not dictated by the user model, or offering a continuous spectrum between complete control and delegation.

4.1.4 Ubiquitous User Modeling

We have seen in the previous Section (4.1.3) how User-Adaptive Systems, such as online shops or travel booking agencies, can learn the users' habits and preferences from their interaction with the system. The more systems we use, the more user profiles are created that reflect different parts of our personality. While they may be directed towards different applications, yet some portions may overlap. For example buying a Singapore travel guide in an online book shop could indicate an interest to travel there, which in turn might help a travel agency to adapt their choice of destinations on their website.

Today, we do not only use a single desktop computer for our activities, but also mobile devices to access services from anywhere. Besides our personal devices, we also interact with information kiosk systems, for example to buy train tickets or to withdraw money from an ATM. In such settings, user modeling is also helpful to adapt the behavior of the services to our personal preferences and situation, such as our native language, knowledge level, cognitive load, and location. For tasks like navigation for example, we may use web-based services at home to browse maps and prepare our route before we enter the car and follow the instructions from the onboard system.

It becomes clear that the benefits of even the most beneficial User-Adaptive System fade considering how many services we use and through how many devices we access them. **Ubiquitous User Modeling** (U^2M) tries to overcome the limitations of individual systems by unifying all user-related assumptions into one consistent model. It means that the user's behavior is constantly tracked at any time, at any location and in any interaction context. Furthermore, the various user models are shared, merged, and integrated on demand.

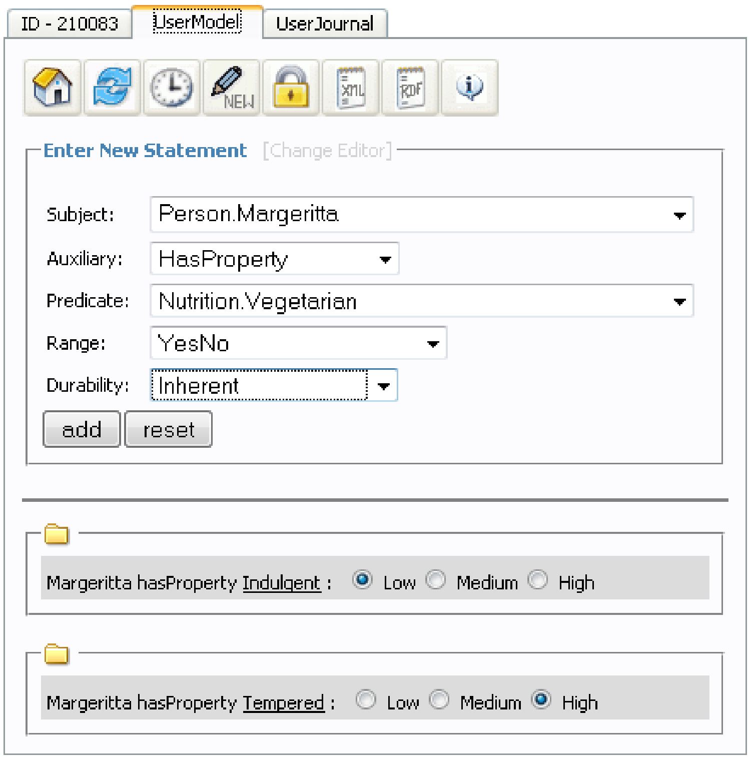
Heckmann (2006) gives the following definition for Ubiquitous User Modeling:

- **Ubiquitous User Modeling** describes ongoing modeling and exploitation of user behavior with a variety of systems that share their user models.

The shift from desktop computing to mobile, situated interaction highly influences the need for decentralized user-adaptive systems. We will introduce the notion of situation or context-aware systems in the next Section (4.1.5). Another aspect of U^2M is the ongoing user modeling with a variety of systems and applications. The challenge is to enable user-adaptive applications to exchange partial user models with each other. These systems are already connected to the World-Wide Web but they are yet isolated. This requires the semantic integration of the distributed heterogeneous partial user models. Finally, the distributed user models need to be merged, synchronized, and fused.

Heckmann's (2006) *UbisWorld*[43] system (see Section 2.4.4.2) employs ontologies to encode information about places and users in machine-readable statements that can be expressed and exchanged using Semantic Web languages like RDF (Resource-Description Framework). *UbisWorld* can be used to represent some parts of the real world, like an office, a shop, a museum, or an airport environment. It represents persons, things, and locations, as well as times, events, and their properties and features. The *UbisWorld Ontology* defines classes and predicates, such as a taxonomy of physical things, places, user characteristics, and activities. *Situational Statements* describe the state of the world in sentences made of a subject, predicate, and object. The vocabulary for the statements is provided by the ontology through concepts, instantiated individuals, and relations. It can easily be extended to specific domains, so the concept of situational statements provides the flexibility that is needed to model any situational context that might be relevant to a specific application.

From the user's perspective, it is crucial to have insight into the user model and to control privacy levels. The *UbisWorld* server provides a comfortable Web-based interface to transparently access the properties of any modeled object within the environment. Figure 47 shows the properties of a sample user *Margeritta*. In the upper section, the Web form allows her to enter new properties as statements, by using the *UbisWorld Ontology*. In the shown example, she is going to express that she is a vegetarian. This statement about her nutrition habits is set by her to be public and may be used by commercial services, like a shopping assistant. The durability of this statement is set to inherent, because it is unlikely to change. Besides, the interface gives a listing of her modeled properties. For example, the model says that she's not a very indulgent, but highly tempered personality. By using the lock icon from the toolbar, Margeritta could hide these properties from the public.



Subject: Person.Margeritta
Auxiliary: HasProperty
Predicate: Nutrition.Vegetarian
Range: YesNo
Durability: Inherent

Margeritta hasProperty Indulgent : [✓ Low] Medium High

Margeritta hasProperty Tempered : Low Medium [✓ High]

Figure 47: Margeritta's user profile in *UbisWorld*.

[43] UbisWorld website: http://www.ubisworld.org/

4.1.5 Adaptation to Situational Context

We have seen in the previous section how User-Adaptable Systems can improve their user interface through learning the habits and preferences of an individual user. In mobile systems however, it is desired that also the content, functionality, and services are adapted to the actual situation in which the mobile device is used. The designer of a mobile computing device has to consider the nature of the context in which interaction takes place. This is especially true for wirelessly connected devices that depend on the network and computational infrastructure of their environment.

4.1.5.1 *Context-Aware and Resource-Adaptive Systems*

Context-aware computing was first discussed by Schilit and Theimer (1994:24):

- **Context-aware computing** is the ability of a mobile user's applications to discover and react to changes in the environment they are situated in.

In the context of mobile distributed computing, they use the term context-aware computing to describe software that uses location information

> *"[...] to adapt according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time."* (Schilit and Theimer, 1994:24)

The question is which factors contribute to the context of mobile human-computer interaction and which can be ignored. A quite specific definition of context for interactions that involve mobile communication is given by Dix (2000) who suggests the following taxonomy of context:

- **Infrastructure**: network bandwidth, reliability, and display resolution.
- **System**: other devices, applications, and users.
- **Domain**: application domain, style of use, and identification of user.
- **Physical**: physical nature of the device, environment, and location.

Each of the different contexts represents a different part of the design space within mobile systems must be placed. The services offered by mobile applications that use wireless communications are a product of the device itself and the supporting infrastructure. The infrastructure context reflects the uncertainty factors that are caused by the dependency of the network. The user interface must be designed to cope with situations of low bandwidth and poor reliability.

System context reflects the distributed nature of mobile devices, as their functionality is spread across a larger system that consists of other devices and users. Devices which are used close to each other can lead to emergent behaviour or interaction, but they might also adversely affect each other.

The designer of mobile applications also has to consider the semantics of the application domain, and design needs to explicitly identify the nature of work being supported. The style of use needs to be adapted to the limited interaction facilities of mobile devices. Depending on the domain, information and services may be associated with individual users.

The nature of physical context can be explained by the example of a modern car radio. It automatically retunes as the car travels between local radio areas and chooses between two antennas to avoid local interference patterns. Depending on the radio area, it provides location-based traffic information. It further adapts the display brightness to day and night conditions, and raises the volume in order to compensate for noise caused by the speed of the car. For most situation-adaptive systems, the physical location within the environment is the predominant aspect of context, so we will discuss it in more detail in the remainder of this section.

Whereas the definition by Dix (2000) is focused on mobile devices, Dey (2001:5) gives a more generalized definition for context, which has been widely adopted in the Ubiquitous Computing community:

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

This definition is intentionally very general, so that it allows the designer of any mobile application to enumerate and determine which information about persons, places, and objects should be considered as relevant for the interaction. Dey (2001:5) also gives the following definition based on his definition of context:

- 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.

According to Dey, there are three categories 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;
- **tagging** of context to information in order to support later retrieval.

Wahlster and Tack (1997) introduced the concept of **resource-aware processes** that adapt their behavior according to varying resources in a rational manner. Based on this concept, Kray (2003:19) describes **resource-adaptive systems** as artificial (and natural) systems that are aware of what resources are available to them at any given time. Such systems should react to changes in the current resource situation.

It should be mentioned that the term *resource* can refer to both: i) cognitive resources of the user; and ii) technical resources of the system. The cognitive resources can change with the flow of activity, and depending on circumstances, like a distraction from the primary task, or stress, caused for example by time pressure or disorientation. In the context of navigational tasks, Baus et al. (2005) identify the following factors as possible limitations to the working memory: i) Time restrictions; ii) Domain knowledge; and iii) Familiarity with the presentation type. Technical resources cover all types of limitations of the presentation platform. In the context of navigation systems, Baus et al. (2005) identify three subtypes: i) Quality and precision of the positional information; ii) Restrictions of the output medium; and iii) Restrictions of the graphics generation process. Technical resources can also be changing, for example mobile devices depend on battery power and network bandwidth. Especially in Ubiquitous Computing environments, the availability of sensors might change.

Wahlster and Tack (1997) distinguish further between *resource-adapted processes*, which have been statically adapted to known restrictions, and *resource-adaptive processes*, which employ a predefined adaptation strategy to perform well when faced with such a situation of varying resource availability. *Resource-adapting processes* address this issue by dynamically generating adaptation strategies or by switching between several ones.

4.1.5.2 *Location-Adaptive Systems*

According to Dix (2000) a mobile device needs to be able to find answers to the following questions in order to adapt itself to its location:

(1) Where am I?

(2) What else is nearby?

(3) How should I behave in the light of (1) and (2)?

Finding an answer to question (1) requires a method to localize the mobile device within its physical environment. The position can be either determined by measuring the distances (trilateration) or angles (triangulation) between the device and two or more points with a known position. Therefore, an arrangement of senders and receivers is required to measure either the runtime or strength of a signal. In outdoor environments, the Global Positioning System (GPS) can be used to calculate the position of the receiving device based on the runtime of radio signals which are emanated from satellites in the orbit. Indoors, this signal is too weak and reflections lead to multiple signal echoes, so that the position cannot be accurately measured. A couple of alternative solutions for indoor environments based on signals like WLAN, ultrasonic noise, or infrared light have been designed to overcome this restriction; a comprehensive survey is given by Hightower and Borriello (2001).

According to Schwartz et al. (2005) these systems can be distinguished by the role of the mobile device as sender or receiver of the signal. In the former case, the so-called **exocentric** localization, the user is actively sending information to the environment which is received by an installation of multiple fixed devices. Some centralized server (e.g. a GSM provider) uses this data to calculate the mobile device's position. In other words, the user is tracked and the privacy may be violated. In the latter case, the **egocentric** localization, the user receives information from senders that are installed in the environment and the user's personal mobile device uses the data to calculate the current position. In that way (e.g. GPS) the position information can be kept private to the user.

Answering question (2) requires a world model for ubiquitous computing. Dix (2000) points out the difference between the actual space and represented space: in practice, a device does not access the "actual" location of itself or other objects directly; it rather accesses some computational representation of location held by itself or some sort of location service. Suitable location models have been introduced in this thesis in Section 2.4.

Question (3) deals with the adaptation of the user interface and the generation of location-aware presentations. For adaptation, rule-based inference systems can be applied. Regarding the generation and presentation of location-aware content, Baus et al. (2002:16) distinguish between two different kinds of location sensitivity for mobile devices: **Active**

location sensitivity, on the one hand, implies that the mobile device is aware of its location which is measured by some kind of positioning system, and automatically generates a presentation to comply with the user's requests. Typical examples are GPS-based car navigation systems that generate and output instructions for the driver based on their self-measured position. **Passive location sensitivity**, on the other hand, requires the mobile device to passively present localized information that it receives from senders in its local environment, e. g. location-based information on traffic status via Radio Data Service (RDS). Baus et al. (2002) suggest that passive location sensitivity might be more appropriate for small devices with limited memory and processing performance, since it moves the burden of adaptation from the device to the environment. The receiving device does not require any intelligence at all, as the radio traffic information example shows.

4.2 From Manual Operation to Proactive Assistance

The mode of interaction with systems can range from completely manual operation to fully automated, unsolicited assistance. This distinction also applies to User-Adaptive Systems, which we have seen in the previous section (4.1): some systems just suggest adaptations, depending on the users' individual behavior, whilst interface agents immediately act upon events like incoming emails or changing context like moving. We can distinguish four levels of assistance:

- Systems without assistance that rely completely on the user to solve a problem;
- Assistance that directly supports the user's goals;
- Assistance that is automatically provided;
- Assistance that is completely invisible to the user.

In the first case, it is completely up to the user to recognize and solve a problem. In case of a breakdown situation, such as being disconnected from the network, the user has to manually recover by taking the necessary configuration steps. Typically, the user has to navigate through menus and modify the system settings which might require to read the manual or to consult the network operator. Nearly all contemporary mobile devices belong to this category. The second class of interface allows the user to directly formulate their needs (e.g. "I want to connect with the internet") and the system will take care of it - or at least guide the user through the necessary steps. The third class automatically recognizes the needs of the user, such as recovery from breakdown situations, and proactively offers help. Finally, assistants can fix problems autonomously and invisibly to the user. Examples are the GSM roaming services which automatically connect with a new provider in foreign regions.

According to Norman's (1986) **approximate theory of action**, the process of performing and evaluating an action can be approximated by seven stages of activity, which are depicted in Figure 48. Path A presents the traditional dialogue between the human user and the machine interface, which relates to the first level of assistance above. Path B shows the structure of goal-based interfaces, which assist their users by assuming the stages of action specification and execution. Path C shows the concept of implicit interaction, where the system tries to recognize the intentions of the user from the current situational context. This allows for fully automatic and even invisible assistance. We will now discuss each of the three paths A, B, and C in more detail.

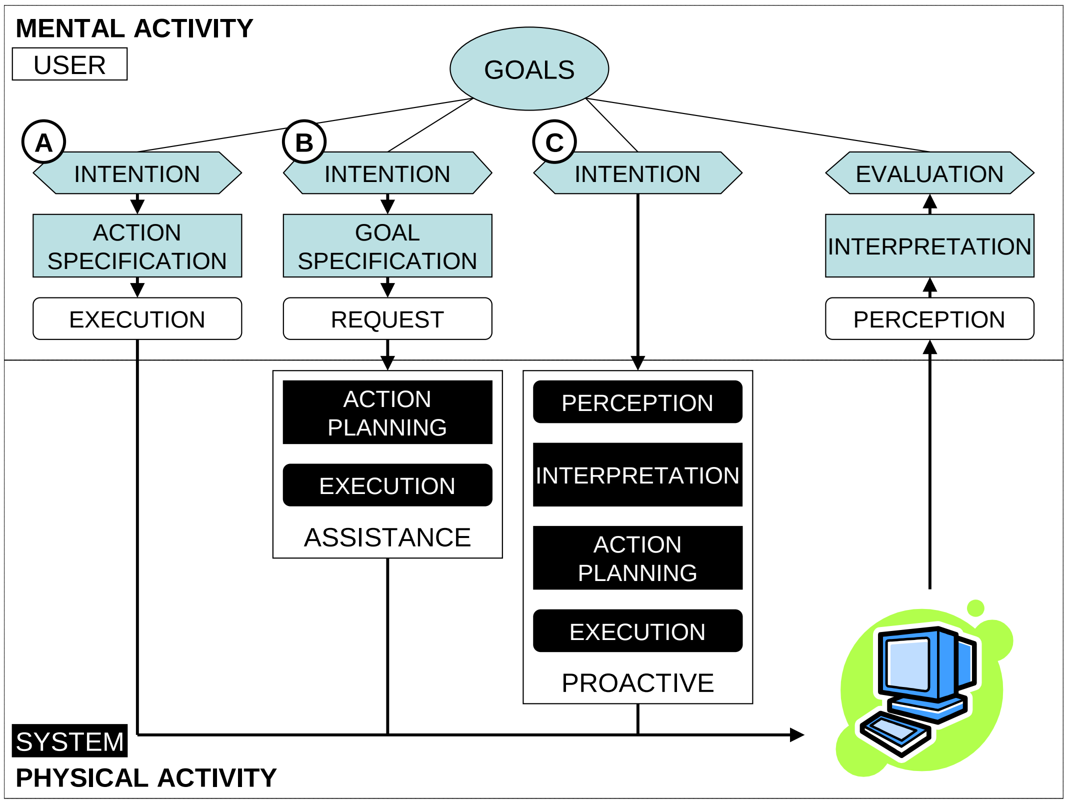


Figure 48: Stages of user activities involved in the performance of a task: A) traditional human-computer interaction; B) user assistance with goal-specification; C) proactive assistance systems based on implicit interaction and plan-recognition

4.2.1 Seven Stages of User Activities

In Figure 48 the path A shows the human-computer interaction model according to Norman (1986). The primary stage is the establishment of the goal. In order to carry out an action, the user has to form the intention, plan an adequate action sequence, and execute the action by using control elements or menus to select and parameterize functions provided by the device. When the functions are executed by the system, they cause an effect: text is formatted in a document, a phone call is initiated, or the room temperature changes. Three complementary steps are required to assess the effect of the action: the user firstly perceives and interprets the altered system state, and secondly evaluates the interpreted state with respect to the original goals and intentions. Eventually the user needs to formulate new intentions, so the cycle continues. In real practice, the stages can appear out of order, and some can be skipped. Instead of starting with the goal, an iteration of the cycle may also start with the perception of a change of state, such as a failure alert originating from the system. In the case of car navigation, there would be two complementary cycles: the user interacts with the car to put the instructions into action, and the system is implicitly responding to the maneuvers by giving new instructions to the user.

This model has been inspired by cognitive sciences, and is today the predominant theory of human-machine interaction with desktop computers. In most everyday appliances, like refrigerators, DVD players, and mobile phones, function-oriented design are realized according to the paradigm of direct manipulation; every function or setting has a one-to-one mapping to a button or menu. However, as the complexity of things has grown, it has already reached its limit. It has become increasingly difficult for the user to be familiar with all the necessary functions and their corresponding buttons in order to get the desired results from a system, especially in infrequently occurring situations. In many areas, such as home entertainment or mobile phones, this concept has already exceeded the cognitive resources of most users. Typical negative examples are TV remote controls with more than 50 buttons. People just do not understand how to configure the devices according to their individual needs, and even the adaptation strategies presented in the previous section have to be applied with great care, as they may compromise other usability goals.

4.2.2 From Function-Oriented to Goal-Based Interaction

Figure 48 B) shows a more cognitive adequate model than A), which allows the user to directly specify their goals and intentions; the user is interested in the effect or result of using a device, not in the functions that are needed to achieve it. This constitutes a paradigm shift from function-oriented towards goal-based interaction. The system assists its user in taking over the steps of planning and executing a sequence of actions, and frees the user from the burden to know about and perform each operation and setting separately.

It is of crucial importance that the user is able to directly formulate his needs and goals in a way that they are understood by the system. Conventional, unimodal, menu-based dialogue structures with a fixed vocabulary are no longer adequate. Instead, conversational dialogue structures with unrestricted vocabulary are needed. The goal would be the use of natural language; however there are still huge technical difficulties to overcome.

Wahlster et al. (2001) have introduced this new interaction concept in the scope of the *SmartKom* project as **situated delegation-oriented dialog paradigm** (SDDP), in which the user delegates a task to a virtual communication assistant that is visible on a graphical display. Complex intentional goals, which cannot be formulated in a simple command-and-control style, can be elaborated in a collaborative dialog between the user and the agent. The Agent is visualized as an anthropomorphous life-like character named Smartakus. *SmartKom* overcomes the difficulties and limitations of natural language through a **symmetric multimodal dialog** system that combines speech, gesture, and mimics (facial expressions of joy or frustration) input and output. It supports the situated understanding of possibly imprecise, ambiguous, or partial multimodal input and the generation of coordinated, cohesive, and coherent multimodal presentations. According to (Wahlster, 2003:2) symmetric multimodality means that all input modes are also available for output, and vice versa. Such a dialogue system must not only understand and represent the user's multimodal input, but also its own multimodal output; *SmartKom* maps its communicative intentions onto a coordinated multimodal presentation. Three versions of *SmartKom* have been developed: i) *SmartKom-Public* is an information kiosk that allows users e.g. to book cinema tickets via a large touch screen and supports the selection of seats through pointing gestures, ii) *SmartKom-Mobile* provides Web access and navigation services on a PDA, and iii) *SmartKom-Home* implements a portal to consumer electronics and communication devices in personal everyday life environments, which assists their users in achieving their goals through a centralized, voice-based assistance system.

Heider and Kirste (2002) call this assisted mode of interaction the **ecological level** of user interface design, where the assistance system (computer) acts as a **mediator** between the user and the devices in their personal environment, as shown in Figure 49. On this level of abstraction, the users perceive the complete set of devices as a single system that helps them reaching their goals. This changes the domain of discourse from the system's view of the world to the user's view of the world.

The prime objective of the *EMBASSI* project (Herfet et al., 2001; Kirste et al., 2001) has been to make the goal-based interaction paradigm available in our everyday life. The acronym *EMBASSI* stands for "Elektronische Multimediale Bedien- und service ASSIstenz", which is the German translation of the project "Multimodal Assistance for Infotainment and Service Infrastructures". Application scenarios for *EMBASSI* are set in various environments, e.g. at home (entertainment and home theater), in the automotive field (danger warning), and in public terminals (ticket sales). The stages of interaction are implemented in the *EMBASSI* system architecture as follows: first, multimodal utterances of the user are processed and translated into goals. Physical interactions of the user are translated into unimodal events. In the next step, multimodal events are aggregated into sentences. The dialogue manager is responsible to translate these sentences from the syntactic level to goals on the semantic level, so that assistance components can develop strategies (plans) to fulfill the goals. A crucial aspect for the planner is to consider the effects of functions across multiple devices in a system. Finally, execution components are responsible for the scheduling and sequential execution of the individual steps in a plan on physical devices. This is mostly a Pervasive Computing problem (2.1.3.4.5).

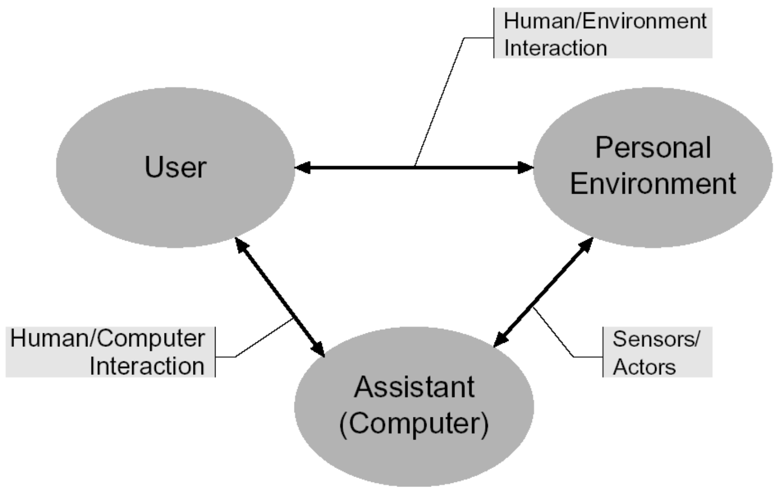


Figure 49: The ecological interface, according to (Heider and Kirste, 2002:597).

4.2.3 Proactive Assistance

The prior modes of function- and goal-oriented interaction, as illustrated in Figure 48 A and B, are focused on the user interface of a particular computing device, where the users tell the computer what they expect the computer to do. We will consider this type of interaction as explicit interaction. The drawback is that the computer is inevitably in the center of the activity and distracts the attention of the user from its environment. This contrasts the paradigm of Ubiquitous Computing where the computational devices should disappear into the background of our attention and become "invisible" (see 2.1.3.4.2).

According to Schmidt (2002), the realization of these visions can only be achieved when parts of the human-computer interaction are transparent and not explicit.

Schmidt (2000:2) defines Implicit Human-Computer Interaction (iHCI):

- **Implicit human-computer interaction** is an action, performed by the user that is not primarily aimed to interact with a computerized system but which such a system understands as input.

For completeness, Schmidt defines implicit output in analogy to implicit input:

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

The basic idea of implicit input according to Schmidt (2002) is that the system can perceive the users' actions in the physical environment and also the overall situational context. This knowledge is then considered as an additional input to the computer while doing a task.

A similar definition has been introduced by Dix (2002:2):

- **Incidental interaction** is where actions performed for some other purpose, or unconscious signs, are interpreted in order to influence/improve/facilitate the actors' future interaction or day-to-day life.

Dix (2002) further distinguishes between expected and unexpected system behavior. For example, automatic sliding doors are expected to open if we try to enter a building (otherwise we would stop in front of them). Hence, their behavior is 'less' incidental than the automatic light inside a fridge that is turned on if we open the door. The light could be considered truly incidental since a naive user might never notice that it is ever turned off to save energy when we shut the door. However, the borderline between incidental, expected, and intentional interaction is fluid:

> *"Comprehension: If users notice, consciously or unconsciously, that incidental interactions are taking place they may come to expect or rely upon them. [..] Co-option: When I know that something will happen when I perform an action, then I may deliberately perform the action to cause the effect."*(Dix, 2002:3)

The concept of proactive user assistance originates from Maes (1994). Maes (1994:33) has defined **interface agents** as follows:

- **Interface agents** are computer programs that employ Artificial Intelligence techniques in order to provide assistance to a user dealing with some computer-based task(s). These agents learn how to assist the user by (i) observing the user's actions and imitating them, (ii) receiving user feedback when they take wrong actions, (iii) being trained by the user on the basis of hypothetical examples and (iv) asking other agents (working for other users) which have more experience for assistance.

Maes' (1994) interface agents are inspired by the metaphor of a personal assistant who is trained to act on behalf of the user. They are especially suitable for tasks which involve substantial amount of repetitive behavior that potentially differs between users. The key qualities for agents to be adopted by their users are competence (knowing when to help the user, how to help the user, and by which means) and trust (to guarantee that the user feels comfortable in delegating decisions). Maes (1994) has implemented interface agents which assist a user with tasks, such as the sorting electronic mal, scheduling of meetings, and filtering of news articles. The email agent *Maxims* (Lashkrai et al., 1994) for example is integrated into the user interface of an email application, and observers the user as the user deals with email. Maxims learns to prioritize, delete, forward, sort, and archive emails on behalf of the user. As the user performs actions, the agent memorizes situation-action pairs of features. When a new situation occurs, such as a new email arrives, the agent tries to predict the user's action based on prior situations. The system computes the distances to learned situation-action pairs and, if it finds a match, it suggests the learned action to the user, if the confidence value is above a threshold. Otherwise the user is not bothered and the system validates its decision based on the action of the user.

The word **proactive** is defined according to Microsoft's Encarta[44] dictionary as:

- taking initiative: taking the initiative by acting rather than reacting to events

According to Wahlster (1999) **proactive interface agents** try to:

- anticipate the user's needs;
- adopt the user's goals;
- provide unsolicited comments.

The concepts of implicit and incidental interaction can be ideally combined with proactive behaviour to implement systems that automatically assist their users without drawing the attention to the user interface itself. The corresponding stages of implicit interaction are depicted in the process model in the Figure 48 above as path C. The users of such a proactive assistance system do not even need to think about how to state their needs and intentions to the computer. Instead, the system uses sensors to perceive their actions in the physical environment and tries to interpret them with respect to a set of possible goals and intentions. Based on the hypothesis about the users' goals, the system tries to proactively assist its user. The system can pursue two different strategies: i) provide information in order to passively influence the user of which actions to take next, considering actions that reduce cost, improve the efficiency, or prevent emergency situations, or ii) actively execute such actions on behalf of the user. However, the latter strategy requires a certain degree of control over the physical environment, as well as trust.

In the next Sections, we will present two different approaches how to interpret the user's intentions from sensor data: plan/activity recognition (4.2.3.1), and rational agents (4.2.3.2).

4.2.3.1 *Plan Recognition*

Plan recognition refers to the process of inferring a plan that is intended to achieve some goal on the basis of the observed actions. This requires that all possible plans that a user could pursue are modeled as a sequence of actions, which the system can recognize from explicit or implicit interaction with the user. According to Jameson (2006:31), plan recognition can be useful to give assistance in routine tasks, so that the system can offer to repeat the user's plan multiple times, but also for help systems that may point out problems with the user's current plan or remind the user of required actions.

Schneider (2009) points out that in the specific case of instrumented environments, sensor data can only partially represent the state of the physical world and that the sensors are

[44] http://de.encarta.msn.com/ visited 23.07.2009

subject to economical and ecological constraints. Thus the challenges are to: i) recognize the actions of the user based on partial information about the world, and ii) select sensors carefully to save resources. Schneider (2009) proposes a state-aware plan recognition model that supports a utility function for sensor selection. The model is based on the concept of a *plan execution state space* which contains an *idle state* that represents no user activity and a set of *goal states* that correspond to the completion of a certain plan. The user executes actions in order to move from the idle towards a desired goal state. Given such a representation, the problem of plan recognition is to estimate the user's current plan execution state given a sequence of sensor observations. Based on the most likely state, the system chooses its next action from a set of *support actions* so that the *expected utility* for the user is maximized. Schneider (2009) applies its system in a smart kitchen environment in order to recognize cooking receipts based on the actions of the user.

Activity recognition can be conceived as a special case of plan recognition, where each plan has only a single goal state that refers to one specific activity. Activity recognition is often applied to the context of Ambient Assisted Living projects in order to assess the health state of elderly subjects based on their behavior, especially with respect to Activities of Daily Living (ADL) (Katz et al., 1963). Such automatically gained information can help to reduce the need for routine visits through care givers, but also indicate emergency situations. Storf et al. (2009) developed an activity recognition framework that describes activities as a continuous sequence of events which is measured by different sensing systems. Complex activities, such as the preparation of a meal, are decomposed into atomic activities. The system is implemented as a multi-agent architecture that is based on *detection agents* which receive low-level sensor data and *facts* from other agents and aggregate them on-the-fly to higher-level facts. The agents also consider the temporal relationship between facts, such as opening and closing a drawer or turning a light on and off. The goal of the project is to detect changes in the behavioral patterns of a subject in order to predict changes in the state of health, which could lead to emergency situations in the near future.

The presented methods, plan and activity recognition, model user activities as plans that comprise a sequence of actions, which can also be hierarchically structured in the case of the framework presented by Storf et al. (2009). This approach is comparable to our proposed activity model (see Section 1.4) that we have derived from Activity Theory (Section 2.2). Both models represent activities as a sequence of actions that are motivated by goals. Activity Theory takes a psychological perspective; the three levels of activity, action, and operation semantically reflect the structure of the human mind, i.e. they distinguish between conscious actions and internalized operations, whereas hierarchic plans are based on the mathematically abstract concept of recursion. Hence Activity Theory is better suited for the modeling and designing of HCI processes, whilst plans are better for the implementation of algorithms. Plan recognition also generalizes from human actors to agents, which includes software systems.

4.2.3.2 *Rational Agents*

The process of perception, interpretation, and action specification, as shown in Figure 48 B) and C) above, can be further refined by the **belief-desire-intention** (BDI) model of rational decision making which has been applied by Wooldridge (2000) as a framework for the implementation of rational agents. According to Wooldridge (2000:2), **Rational Agents** (RAs) have the following properties:

- Situated – RAs are embodied in an environment, are capable of sensing their environment, and have a repertoire of possible actions that they can perform in order to modify their environment;
- Autonomy – each RA has its own beliefs, desires, and intentions;
- Proactiveness – RAs have goals that they try to achieve (goal-directed behavior);
- Reactivity – RAs respond to changes in the environment;
- Social ability – RAs can communicate with other agents (including humans).

The BDI model is focused on the deliberation of intentions, based on the agent's beliefs and desires. **Beliefs** represent the information about the world that is available to the agent. Using the term *belief* - rather than *knowledge* - recognizes that what an agent believes may not necessarily be true. **Desires** represent objectives or situations that the agent would like to accomplish or bring about. **Intentions** represent what the agent has chosen to accomplish in the near future. Wooldridge (2000:31) suggests the following control loop for agents:

- Based on current belief and new perception, the belief revision function determines new set of beliefs about the world;
- The deliberation function takes beliefs and returns a set of intentions, selected by the agent to achieve in the future;
- The planning function decides how to realize the current intentions, based on the beliefs about the world. Plans can be chosen from a library or constructed from scratch.

Wooldridge (2000) considers plans as recipes for achieving intentions; a plan is a tuple consisting of: i) pre-condition (the circumstances under which a plan is applicable), ii) post-condition (what states of affairs the plan achieves), and iii) a body (sequence of actions).

4.2.3.3 *Mixed-Initiative Interaction*

Mixed-Initiative interfaces aim to take advantage of the power of direct manipulation (explicit interaction) and automated reasoning (plan recognition). This approach assumes that intelligent systems and users collaborate efficiently in order to overcome typical problems of agent-based systems, such as poor guessing about the goals and needs of users, inadequate consideration of the costs and benefits of automated action, poor timing of action, and inadequate attention to opportunities that allow a user to guide the invocation of services (Horvitz, 1999).

Lieberman and Espinosa (2006) have developed such a proactive, mixed-initiative assistance system for the home-entertainment domain, called *Roadie*. Similarly to *SmartKom* and *EMBASSI,* it provides a goal-based user interface that abstracts from the function-oriented user interface of the individual devices. If the user manually operates a connected device, for example the user turns on the DVD player, the system proactively consults a plan recognizer that is connected to a database of common sense knowledge in order to infer possible user goals. It then presents the most likely goals in a dialogue box to the user. The user can select one of the goals, for example to record a DVD, and advises *Roadie* to automatically perform the necessary actions to set up the system accordingly. If *Roadie* cannot perform the action itself, it asks the user to perform it, e.g. to connect the DVD recorder to a player. *Roadie* can also explain each of the steps in detail to facilitate learning, and provide assistance in the case of failure. According to the principle of mixed initiative, the user can also directly type in a goal in order to activate *Roadie*.

4.3 Activity-Adaptive Systems

We have already presented how user-adaptive and context-aware systems are able to automatically adapt their appearance and behavior according to the user's likes and dislikes in certain situations. We can conceive that such a system is most likely used in order to pursue certain goals. As we have seen in Section 2.2 about Activity Theory, these goals do not stand alone, but are rather motivated by higher-ranking activities and needs.

We believe that the current definitions of user-adaptive, context-aware, and even goal-based systems lack an adequate representation of the user's activities and needs, and we assume that the addition of such knowledge will be an important step towards seamless user assistance for activities of everyday life.

Based on our definition of a situational semantic activity model (see Section 1.4), we give the following definition:

- An **Activity-Adaptive System** is an interactive system that adapts its behavior to the current activity of the user based on semantic knowledge about operations, actions, and activities in the environment, given by a semantic activity model. In case of a situational semantic activity model, the situational context of the activity allows the system to apply an additional layer of knowledge about rules and plans supporting inference and plan recognition.

Such an activity-adaptive system or service has knowledge about **why** it is used, namely the motivational context of use. In analogy to context awareness, we will term this property **motivation awareness**.

We present a layered architecture for User-Adaptive and Activity-Adaptive Systems in Figure 50. Similarly to the Semantic Web layer cake model (Berners-Lee, 2000), the ontology layer at the bottom provides the technical foundation for user and activity models. The user model is supposed to represent personalized information about a specific user, whereas the activity model should contain general activity patterns that are domain- (and not user-) specific. Based on the user model, adaptation rules describe adaptation processes which can be implemented by User-Adaptive Systems (UAS). These rules typically adapt the user interface and the presentation of content according to the preferences (and sometimes situational context) of an individual user. An optional location model allows for location awareness, e.g. to implement location-based services. On the other hand, plans build the knowledge base for activity- and plan recognition processes that serve as implicit input for activity-adaptive processes. The proactive aspect of interface agents implies that the user permits the system to start a dialogue at any time. Furthermore, in order to act autonomously on behalf of the user, the system requires control over the environment. Transferring decisions and control to the agent supposes trust, which can only be achieved by an explanation component that gives reason for the system's actions. Therefore we include an explanation layer to the implementation of Activity-Adaptive Systems (AAS).

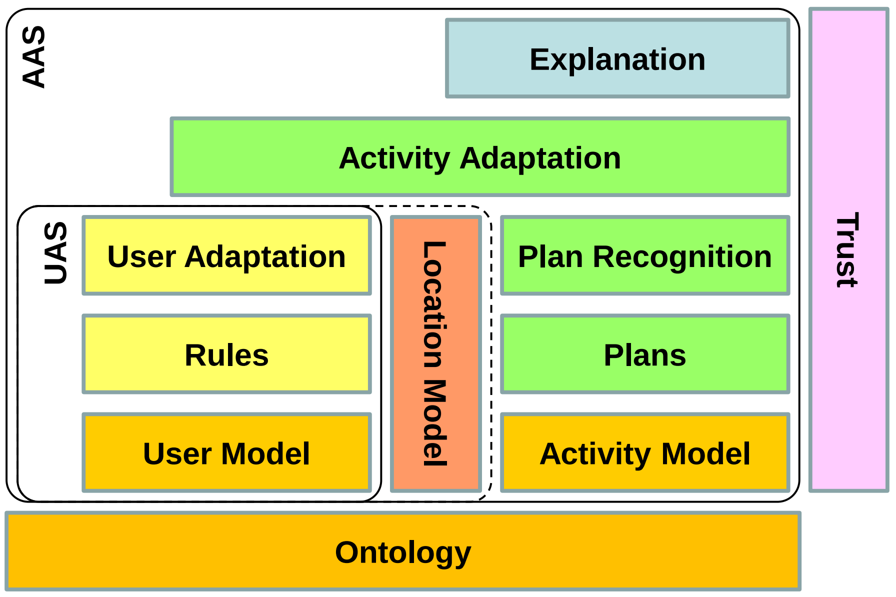


Figure 50: Layer model of User-Adaptive and Activity-Adaptive Systems.

We expect to gain the following benefits from grounding the adaptation processes on semantic knowledge about activities:

- More effective adaptation of the User Interface (UI) and presentation of information according to a deeper understanding of the current activity and motivation of the user;
- Reliable activity- and plan-recognition processes for the implementation of proactive behavior in Ubiquitous Computing environments (based on a situational activity model);
- Chaining input and output between systems according to the flow of activities;
    - Complement history-based suggestions (e.g. recently used documents, applications, or Web sites) with predictions.

Take e-commerce as an example for adaptation, the first aforementioned point. Typical consumer-to-consumer (C2C) trading platforms allow buying or selling activities. Buying of items is generally supported by the browsing of a database, supplemented with a search function. Selling is supported by a dialogue that requires the user to enter a product description and price. However, before the user makes the decision to sell a good, it is typical to estimate its attainable value. Unfortunately this action is rarely supported. As a workaround, one can browse the database for similar items, but the search is optimized for buying, not selling. In their online help, the eBay[45] auction platform suggests their users to manually adapt the search by showing closed auctions. Such adaptations could be automated in the future if the system would be aware that the user is not searching items in order to make a buying decision, but instead to decide about selling something. Furthermore, the system could consider the overall motivation for the user to sell a good, relating to typical questions like *"Is it worth the effort to try to sell this good (or is it junk)?"* or *"Is it reasonable to upgrade to a new version of this good by selling my old one?"*

[45] http://www.ebay.com visited 23.07.2009

By having knowledge about the flow of activities on a semantic level, an Activity-Adaptive System could infer its relationship to other systems or artifacts in a Ubiquitous Computing environment regarding input from preceding actions and output to subsequent actions. Such knowledge could support the exchange of data objects between related applications, like numerical results, addresses, and documents. The chaining of applications is already known from the aggregation of Web Services and could be transferred to Ubiquitous Computing.

Let us illustrate the difference between user-adaptive and activity-adaptive in the case of a calculator. A user-adaptive calculator would hide scientific functions from a novice user, for example. An activity-adaptive calculator could import input values according to the flow of activities, e.g. product prices from e-commerce or distances from a navigation system, and provide results, such as a sum or unit conversion, to other applications like a spreadsheet.

Car navigation is also a good example for the chaining of information: current systems require the user to manually enter the destination address. Horvitz et al. (2007) have shown how the user-adaptive system *Mobile Commodities* (MC) can avoid this dialogue; MC implicitly identifies a destination from the current partial trajectory of the car, considering previous destinations and time of day. Furthermore, the MC system is able to automatically derive destinations from an online calendar. Adding semantic knowledge about background tasks allows MC for opportunistic planning: the system proactively checks if additional goals can be accomplished during the current trip, e.g. to buy groceries or gasoline, and suggests according routes to the user. Hence the MC system can already be considered as being activity-adaptive. We will introduce our own activity-adaptive assistance systems *UBIDOO* and *VISTO* later in this thesis in Section 8.2.2. In our case, the semantic activity model allows a seamless transition from outdoor navigation with *UBIDOO* to indoor navigation with the *VISTO* system. In the case of airports, *UBIDOO* would suggest a parking place that is close to the check-in, and then *VISTO* would guide the visitor indoors to the check-in and gate. In case of a business trip, the car could forward the actual mileage information to a travel expense report application, since this activity typically follows the travel process.

Currently, history-related recommendations are popular that allow shortcuts to recently used objects. Web browsers automatically suggest complete URLs as the user starts typing some letters, based on previously visited sites. In the context of information chaining, activity-adaptive systems have potential to improve recommendations through predictions.

User modeling techniques have already been applied in order to predict what a user would be interested in next according to previous behavior of him/her and others. For example, Zukerman et al. (1999) have presented a UAS that applies a predictive statistical model based on previously visited web locations to preload web pages to reduce waiting time for the user. The system employed a collaborative machine learning approach for its user model, assuming that all users behave similarly. However, in practice, there is no obvious clear objective that applies to all users; some search specific information, while others are just browsing. Furthermore, the sequence of requests observed by the Web server is only a partial record of the user’s movements, since the server can’t observe movements to external locations.

We assume that an activity-adaptive system could overcome these limitations through semantic annotation of the Web pages, i.e. with existing Semantic Web technologies. If the system has semantic knowledge about each page’s functionality with respect to activities and goals, plan recognition methods could be applied and the quality of the predictions would increase.

For example, if the browser application observers that the user has just ordered (but not paid) products from an online shop, the browser could recognize this action as part of an "online shopping with payment by bank-transfer" activity and recommend to visit the user's banking website and direct him to the money transfer page.

There are countless online shops and banking websites, so it would be tedious work to manually define good predictions for each of them. Also, collaborative machine learning would probably fail because of the individual differences between the users regarding their payment preferences and banks. But if only generalized activities, such as online shopping, were explicitly modeled as plans on a high level of abstraction, and if web sites were adequately annotated, so that they can be recognized as corresponding actions, a plan recognition approach might have been feasible, at least for certain domains and use cases.

4.3.1 User Assistance in Intelligent Environments

Whereas User-Adaptive Systems typically reside in the traditional desktop computing environment, proactive systems based on implicit interaction are ideally suited to enhance activity that is situated within the real world. The important steps are to sense the activity of the user, match the observed actions with potential needs and goals, and finally present information in a non-obtrusive way without distracting the user from their current task. Beginning with the Ubiquitous Computing paradigm (2.1.3.2) in the early nineties, much research effort has been spent to computationally-augmented artifacts, sensor networks, context models, and presentation media. Intelligent or Instrumented Environments (2.1.3.4.3) employ sensors and actuators in order to assist their users in everyday situations, for example at home, in museums, or shops. Intelligent Environments provide assistance on the ecological level of networked devices and artifacts, adapt to individual users and their situation, and pro-actively respond to goals and intentions.

In Chapter 5 we will present our former research project *REAL*, which investigated how a proactive system can assist its users in an intelligent environment. We present a comprehensive scenario and two assistance applications which assist their user in navigation and shopping tasks. We will furthermore raise questions about the design and development of such systems.

4.4 Summary

In the beginning of this chapter, we motivated the need for user assistance in the context of Ubiquitous Computing environments and Ambient Assisted Living scenarios, but we also pointed out that new technologies must be easy to learn, usable, and should be aware of the user's goals. Otherwise they will not be accepted by the user, as it was the case with the "Clippy" Office assistant. We introduced the concept of a user model and discussed the paradigm of user adaptation as a basic means to increase the usability of assistance systems. Such User-Adaptive Systems adapt their UI and the presentation of information to the individual user according to the user model. In this context, we referred to the related research domain of Ubiquitous User Modeling (U2M). Then we discussed the adaptation to situational context and reviewed various definitions of context and context-awareness. We also introduced the concept of resource-aware processes. Furthermore we focused on location-adaptive systems and the related question of positioning.

We also investigated three different theoretic models for human-computer interaction, namely Norman's (1986) approximate theory of action that models a function-oriented style of interaction, goal-based interaction on the example of the projects *SmartKom* and *EMBASSI,* as well as the concept of proactive interface agents. In the context of *SmartKom* we presented the concept of symmetric multimodality. Regarding *EMBASSI*, we introduced what an ecological interface is and the paradigm of computer-mediated human-environment interaction. In the scope of this thesis, we are interested in the possibilities of instrumented environments for user assistance. Hence we explained the paradigms of proactive computing, based on implicit interaction and interface agents. Furthermore, we described the related techniques of activity- and plan recognition, the belief-desire-intention framework, and mixed-initiative interfaces.

We closed the chapter with our definition of Activity-Adaptive Systems and the concept of motivation awareness. In this context, we presented a layered architecture for AAS that illustrates the novel aspects of activity adaptation in comparison to user adaptation. Finally, we discussed potential benefits of activity adaptation by some example scenarios.

The presented concept of resource-adaptive processes was the guiding theme for the CRC 378, in which we conducted our research on instrumented environments in the project REAL, which we will report on in the following chapter.

5 User Assistance in the SUPIE Environment

In this chapter we will review the results of our finished project *REAL* (*"Dialogue and Interaction in Instrumented Spaces"*, 2002-2005, CRC 378, Funded by DFG; see Stahl et al., 2004; Stahl et al., 2005) which has been concerned with the research question:

How can a system assist its user in solving different tasks in an instrumented environment?

Instrumented environments consist of distributed computational power, presentation media and sensors and also entail the observation and recognition of implicit user interactions in the environment. This offers the possibility to infer about a user's plans and intentions, and to proactively assist in solving their tasks at hand (see Section 4.3.1). We will now present a comprehensive example scenario that outlines how a traveler at an instrumented airport environment could be supported in their navigational and shopping tasks in the near future. Then we will relate the envisioned scenario to the implemented demonstration prototypes in our instrumented lab environment called *SUPIE* (Saarland University Pervasive Instrumented Environment). Finally, we will explain the architecture and infrastructure of *SUPIE* and introduce the services that are provided for the development of assistance applications.

5.1 The Airport Scenario

Imagine a user, who has just checked in at a large airport, and wants to buy a digital camera at the duty-free zone before boarding the plane. The first goal is to find an electronics shop which offers a good selection of digital cameras. The user activates their PDA to explore the environment, queries the system for electronics shops, and requests macro-navigational aid. Since the user has to carry hand luggage, it is rather inconvenient to look at the display of their PDA, and it feels more comfortable to be guided by personalized directions given by wall-mounted ambient displays.

Upon entering the shop, the user picks up a shopping cart that is equipped with a tablet-PC based shopping assistant. A virtual salesperson appears nearby and gives a short introduction to special sales offers and the shop's departments. Finally, the virtual character moves towards the direction of the shelf and kindly asks the user to follow her.

Facing the shelf with many digital cameras, the user remembers a certain model, which has been recommended by a friend. The user engages in a dialogue with the shelf: "What's the price of the ACME 500 camera?" Suddenly a spotlight appears and directs the user's attention to the camera package, according to our definition of micro-navigation in contrast to macro-navigation (see Section 2.3.3). "I have a special offer for you today, only 399 euro" the shelf replies. As the user picks up the camera, the shelf informs the user about some distinctive features of this model. The user continues the dialogue with questions regarding technical details, and the shopping assistant supplements its verbose answers with visual information that is shown on the shopping cart's display. Additionally, a browser window appears on the plasma display, replacing the advertisement by the product website that is provided by the vendor. As the user picks up a second camera, the system instantly provides product comparison information.

Meanwhile, the alarm manager application has estimated that the user should leave the shop immediately in order to board the plane. The application advises the presentation manager to serve a reminder message with highest priority to the user. The message appears on the same plasma display which has been used before.

The user quickly buys the camera and moves on towards the gate, following the navigational aid that is presented on the public displays along the corridor.

5.2 Navigation and Shopping Assistance in the SUPIE Environment

In the following, we will relate the services that are described in the fictional scenario to the actual assistance applications that we have realized in our instrumented lab environment called SUPIE. Then we present each assistance application in more detail.

In the scenario described above, the user pursues navigational goals inside the terminal building in order to find the shop, and receives assistance through public displays. In order to realize such navigational assistance in our lab, we have developed the indoor positioning system LORIOT, which uses a PDA to receive RFID- and infrared-beacon signals and to process them with dynamic Bayesian networks. The details of this system will be explained in more detail later in Section 5.4. The PDA uses the estimated location in order to trigger the presentation of navigational aid on public displays; Figure 51 A) shows a rear-projection that is situated at the entrance of our lab, and B) shows small ambient displays that are mounted next to the office doors. The user's location can be optionally made available through a ubiquitous user model in order to enable further location-based services inside the shop.

The scenario further describes how the user is guided towards the desired camera model inside the shop by a virtual salesperson. We have implemented this service through a virtual room inhabitant called *Cyberella* (Kruppa et al., 2005; Kruppa, 2006) that introduces the available products and services to the user. Her appearance is depicted in Figure 51 C). If the user is close enough to the shelf, the *SearchLight* (Butz et al., 2004) provides micro-navigational aid that draws the user's attention to the product itself. It projects a bright spot onto the package that helps the user to locate the camera, as shown in D). The virtual avatar and the spotlight are accomplished by a combination of a steerable projector and spatial audio system, as shown in Figure 52.

In the shopping scenario, we explore how to proactively assist the user in achieving their goal of making the best possible buying decision within a limited given time. We employ an RFID-technology based infrastructure of readers and tagged products to sense implicit user interactions, such as picking up a product from the shelf, or putting it into the shopping cart. The *Smart Shop Assistant* (Schneider, 2004) is implemented in the style of a Web application that is running on a server.

As the user interacts with real products on the shelf and the shopping cart, the assistant proactively serves product information to the user in a Web browser, either on a tablet-PC display, which is mounted at the shopping cart, or any other public ambient display. The proactive behavior of both assistants is complemented by mobile multimodal user interfaces running on a PDA.

The *BMW PersonalNavigator* (BPN) (Krüger et al., 2004) application, which was jointly developed by BMW and the DFKI, offers point-to-point navigation in combined outdoor and indoor environments. In addition, it provides an exploration mode, which allows the user to query information on points of interest within a three-dimensional navigational map. The user may formulate their request using combined speech and stylus gestures.

The *Mobile ShopAssist* (MSA) (Wasinger, 2004; Wasinger, 2006) application also runs on a PDA and recognizes interactions with products on the shelf as extra-gestures, which can be mixed with intra-gestures on the touch screen, such as product selection, handwriting, and speech.

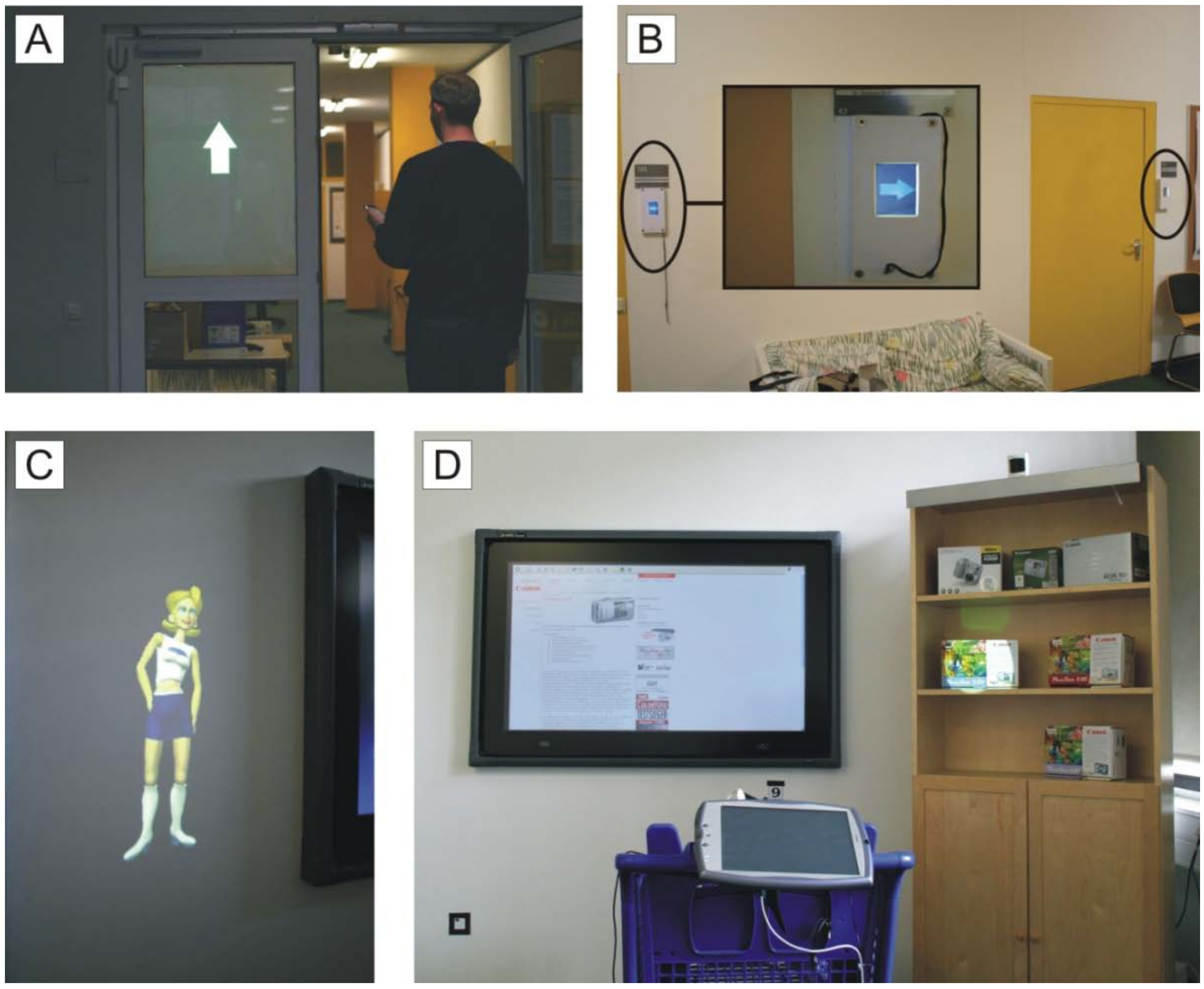


Figure 51: Images taken from our user assistance applications, showing macro-navigational aid (A,B) in the corridor and micro-navigational aid (C,D) in the shopping room.

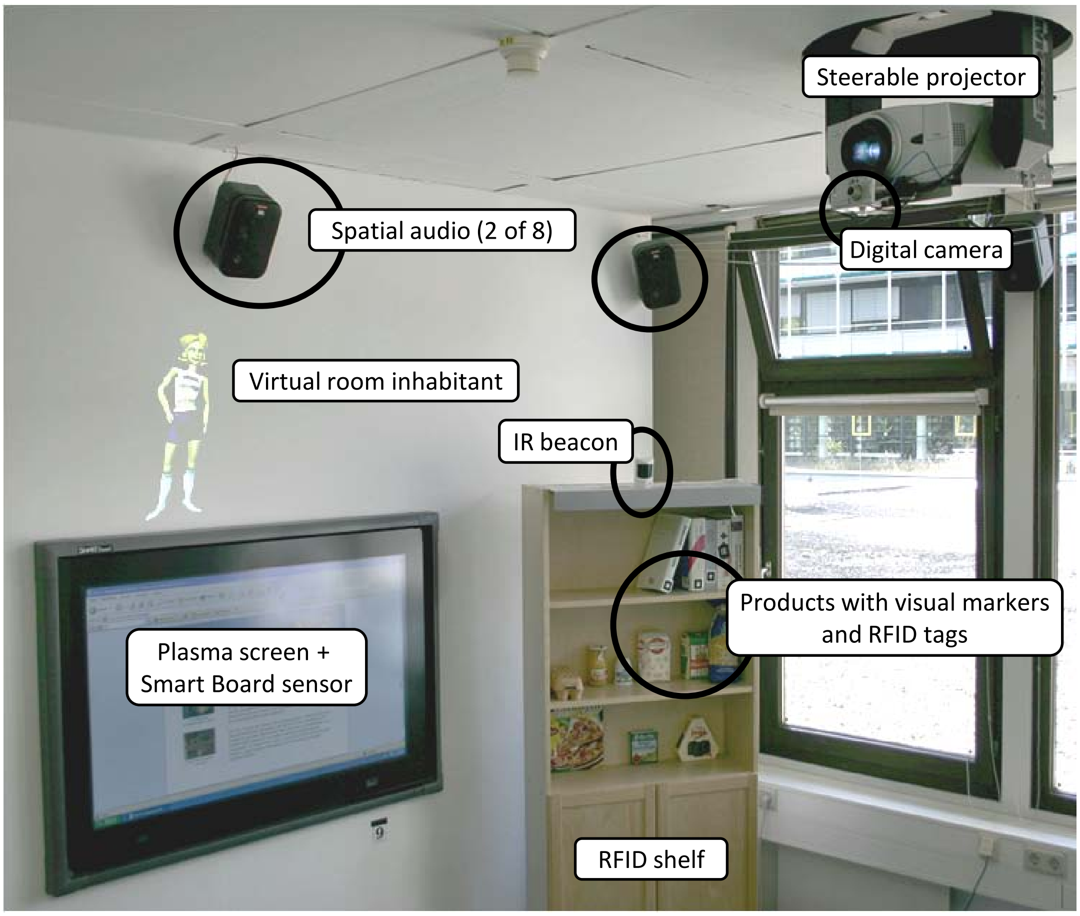


Figure 52: Hardware components of the SUPIE environment

5.2.1 Macro-Navigational Aid: Helping the User to Find a Shop

The *BMW Personal Navigator* application resides on a PDA (Figure 53) that is also used to determine the user's position within the environment, and provides a mobile multimodal interface that allows the user to explore the environment through the combined use of speech and gesture, and to select from a set of routes and destinations.

The motivation behind the BPN project has been to provide a *situated personalized navigation service* (Krüger et al., 2004) that allows travelers to prepare their trips at home and also provides navigational aid, which extends into indoor environments. The BPN system implements an integrated assistance service that seamlessly combines the usage of desktop PC at home, a built-in car navigation system, and a mobile handheld device (PDA).

At home, it allows the user to obtain route directions and maps, to choose personalized events of interest at their destination, to book an appropriate hotel, and to retrieve further information like the current weather. The user can intuitively create a new travel itinerary online by directly choosing an address from the contacts stored in their PIM software (*vcard* format) and a date from an electronic calendar entry (*vcal* format). Route finding and map drawing is then performed through an open-source GIS server. Commercial street data from NavTeq has been supplemented with path networks inside of certain buildings, such as our computer science institute. This has been done manually by editing scanned architectural plans of the building.

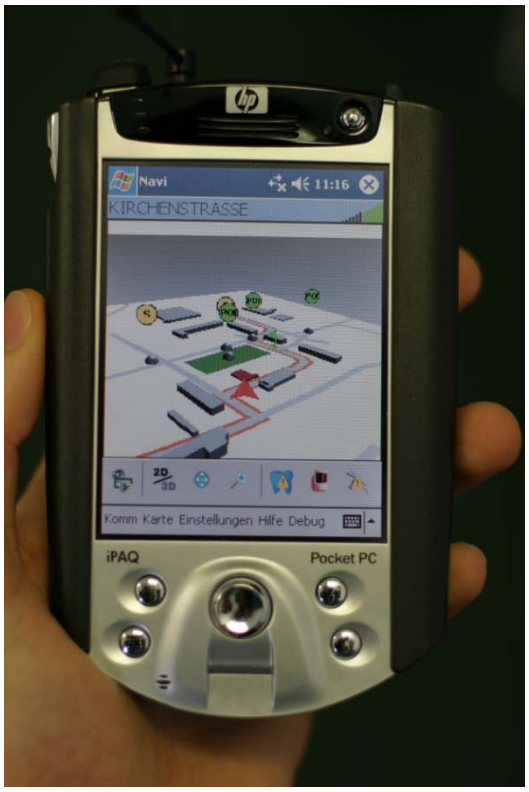


Figure 53: The BMW Personal Navigator.

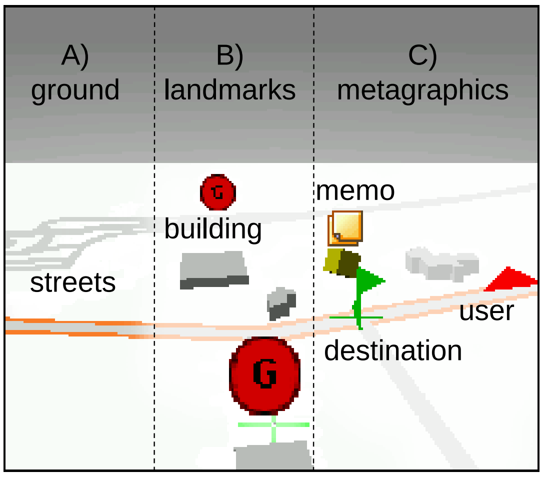


Figure 54: The graphical map consists of 3 layers: A) ground plane, B) landmark objects, and C) metagraphical elements.

In addition to the routing information the system uses Web services to retrieve local information for the destination address, such as the weather forecast and events that match the user's interests. This information is compiled into a travel itinerary and stored on the PDA, which is subsequently used as a connecting link between the home, car, and pedestrian navigation situations.

As the user leaves home and enters the car, the PDA connects to the car's onboard navigation system via a Bluetooth connection and transfers the route information so that it can be conveniently accessed through the car's built-in user interface. The PDA itself is not used for navigation in the car, since the onboard systems are usually superior to PDA's regarding their computing power, sensor accuracy, and user interface ergonomics. As the driver reaches the destination and stops the car, the current geographic position is transferred to the PDA to avoid a 'cold start' situation of its GPS receiver, which can take several minutes. Alternatively, an assisted GPS (A-GPS) unit could be used. Now the PDA presents a mobile user interface to navigate the user into and through the building, which exhibits a rich degree of multimodal user interaction in the form of both user input and presentation output, which we will describe in the following section.

5.2.1.1 *Mobile Multimodal Interaction*

Once activated, the system responds to the user's current estimated position as implicit input and provides macro-navigational aid as the user approaches decision points. The M3I (Mobile Multimodal Interaction) platform is designed to easily support indoor and outdoor navigation tasks, and uses several modalities for user input and the presentation of output

(Wasinger, 2003c). Whereas 2-D/3-D graphics and synthesized speech are used to present useful information on routes and places, embedded speech- and gesture recognition allow for situated user interaction.

The user can refer to a graphical map of the indoor and outdoor environment, which is presented in a 3-D perspective from three selectable views. We use a VRML-based 3-D graphics engine[46] for visualization and interaction with the map that allows the user to zoom in and out. The view can be aligned to match the user's current 'heading'-direction or to 'true north'. The interactive map consists of three layers as shown in Figure 54: The bottom layer A) is the ground plane of the model, which is textured with a street map of the area. It shows all streets, but without any labels. The image is externally generated by a GIS server that has access to roadmap data. In indoor environments, an architectural drawing of the building is used instead. On top of the map is a second layer B) containing landmarks, which are represented by abstract 3-D models. Above the model we place icons in order to indicate the landmark's category. Current categories are G (Navigational Guides) such as tall buildings, and *POI* (Points of Interest). In indoor environments, walls are also shown. The third layer C) comprises additional meta-graphical information. Here we use iconic 3-D models to represent information about the user's position and destination. As the sensed location changes, the user-icon continuously moves on the map. Additional icons are used to symbolize and access geo-referenced multi-media memos, such as voice recordings, or photos taken with a digital camera.

The **navigation mode** directs a user from start to destination through combined graphics and speech output. A flag symbol is used to visualize the next navigational sub-goal on the map. As the user reaches the flag, it is moved to the next sub-goal and an arrow indicates the new direction. The speech synthesizer receives navigational data such as street names, distance, turning angle, and landmark information from the travel itinerary (encoded as XML files), from which it then generates appropriate verbal route descriptions.

The navigation mode is best suited when a user is either under a high *cognitive load* (e.g. business people), or simply uninterested with their immediate surroundings. For situations in which the user has leisure time to spend (e.g. tourists) and has no clear idea of where to go, the **exploration mode** is better suited. For example, the user might have a few hours of spare time in the center of a town. Here the user may desire multiple activities like shopping, sightseeing, and resting in a café. However, the chance of missing an interesting place is high, because the user may be unaware of its existence, or they may not be able to recall where they had seen it. The user can switch between both modes at any time. The exploration mode allows the user to explore the map as the camera's perspective focuses on stylus gesture input. This is in contrast to the navigation mode, in which the camera strictly focuses on the user's current location.

Studies such as (Oviatt, 1996; Oviatt 1997) show that there are definite advantages of combining the use of speech and gesture in the interaction with maps. These results show that users strongly prefer and are also more efficient with multimodal interaction over speech-only map interactions. Hence the BPN user interface implements a media fusion component (Wasinger, 2003b) that allows the user to formulate their request using speech-only, gesture-only, or speech and gesture combined interaction. The latter case is the most

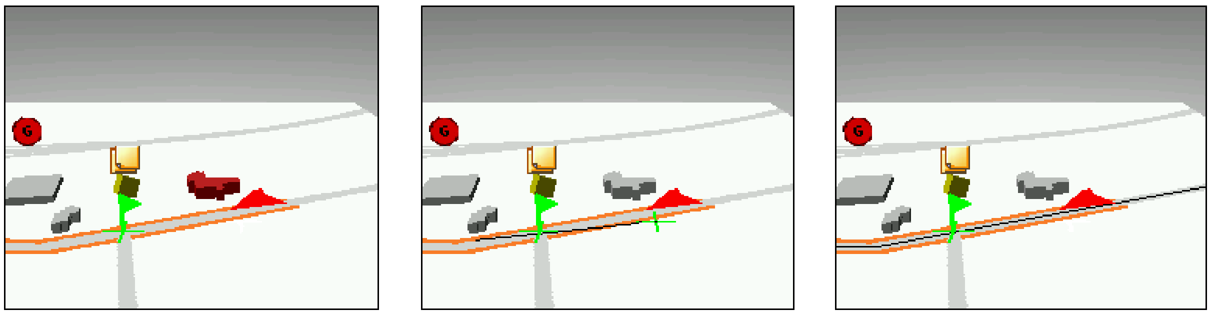
[46] Cortona VRML Viewer for Windows Mobile, Website: http://www.parallel-graphics.com

interesting, where the user verbally formulates their request to the system and refers to the selected street or landmark object through a deictic expression, for example by saying "this" or "here". The basic commands to request information about streets and landmarks using speech and a gesture *G* include "what is this *G* ?" and "describe this *G*". While the user speaks into the microphone, they can use the stylus to tap a landmark or to mark a street. A more complex statement is for example "how do I get from here *G* [..to here *G*] to here *G* ?", where multiple objects can be selected.

For stylus input within a densely crowded map region, it may not be clear as to whether the user's intention was to denote a street, a landmark, or a point on the ground map. The BPN interface solves this problem through the use of gestures for disambiguation. A freehand line can be drawn onto the map with virtual ink, which offers various possibilities of interpretation. The current implementation differentiates between pointing gestures to denote landmarks (see Figure 55 A) and slide gestures to denote streets (see Figure 55 B). In order to give the user some direct feedback, the matched landmark or street is highlighted, please see Figure 55 A) and C) for an example.

Speech input and output is provided by IBM's 'Embedded ViaVoice' text-to-speech synthesizer and speaker-independent recognizer. We incorporate both a set of static (pre-compiled) grammars, and a set of dynamically created grammars. The static grammar covers map control functionality (e.g. "zoom in"), trip queries (e.g. "where is my start?"), and simple multimodal interaction with objects (e.g. "what is that *G*?"). The dynamic grammars are created each time a map is loaded and allow for interaction with object 'types' such as streets and landmarks (e.g. "what is the name of this *G* church?"), interaction with object 'names' (e.g. "take me to Poststraße"), and detailed interaction with the individual landmarks (e.g. "what are the opening hours for this *G* museum?").

In (Wasinger, 2003) we have also presented a strategy to adapt graphics (level-of-detail) and speech (pitch and prosody) output according to the user model, such as age, preferences, cognitive load, and situational context.



A) Selecting a Landmark with a pointing gesture.

B) Drawing a freehand slide gesture over a street

C) The matched street is highlighted.

Figure 55: Different types of stylus gestures

5.2.1.2 *Presenting Navigational Aid on Ambient Displays*

The user of the BPN is also supported by small ambient displays, which we have mounted to the wall next to the regular doorplates of five offices, similar to the *HERMES* project (Cheverst et al., 2003). As the user approaches these displays, arrows are presented that guide the user towards their current destination. Figure 56 shows an example route through our lab, where three displays are used to present arrows, and one display acts as door display. The ambient door displays have the advantage of being fixed to the environment so that their position and orientation is always precisely known, whereas the position of the mobile device can only be roughly approximated. Especially sensing the orientation of the PDA is error prone, since magnetic compass devices are often irritated by electromagnetic fields inside of buildings and can give misleading directions. Hence the spatial information presented on the wall-mounted displays is more accurate and reliable than similar graphics on the mobile screen.

The ambient displays are triggered by the BPN handheld device through virtual proximity sensors. When the distance between the estimated position of the user and the location of the display falls below a certain threshold, the PDA continuously sends presentation requests for the according arrow graphics via a wireless LAN network to the ambient display. If the user moves away from the display, it switches back to its default mode of operation, which shows an interactive door display application.

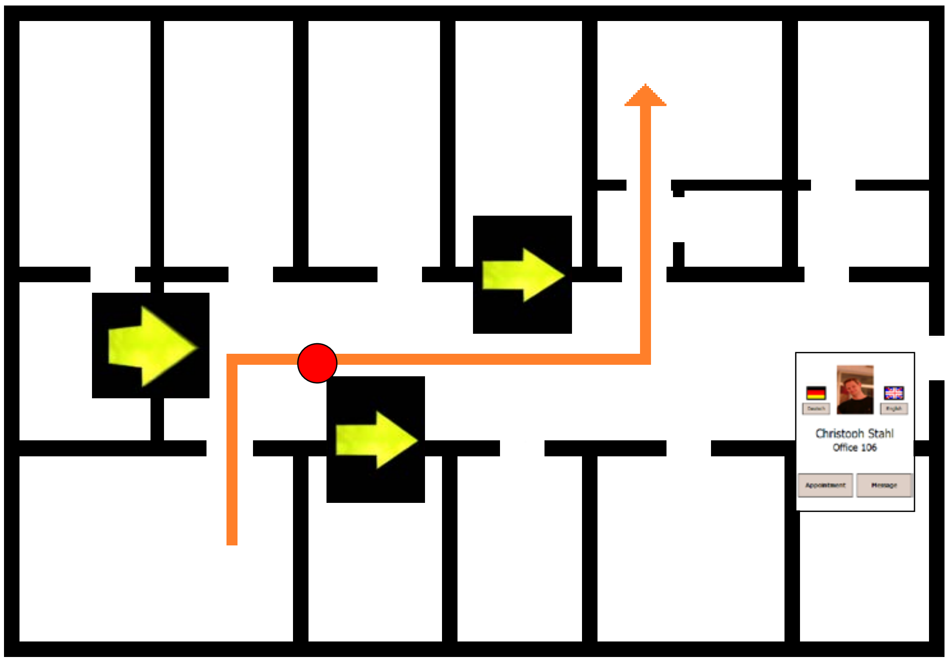


Figure 56: The handheld device connects to ambient displays in the environment.

5.2.2 Shopping Assistance

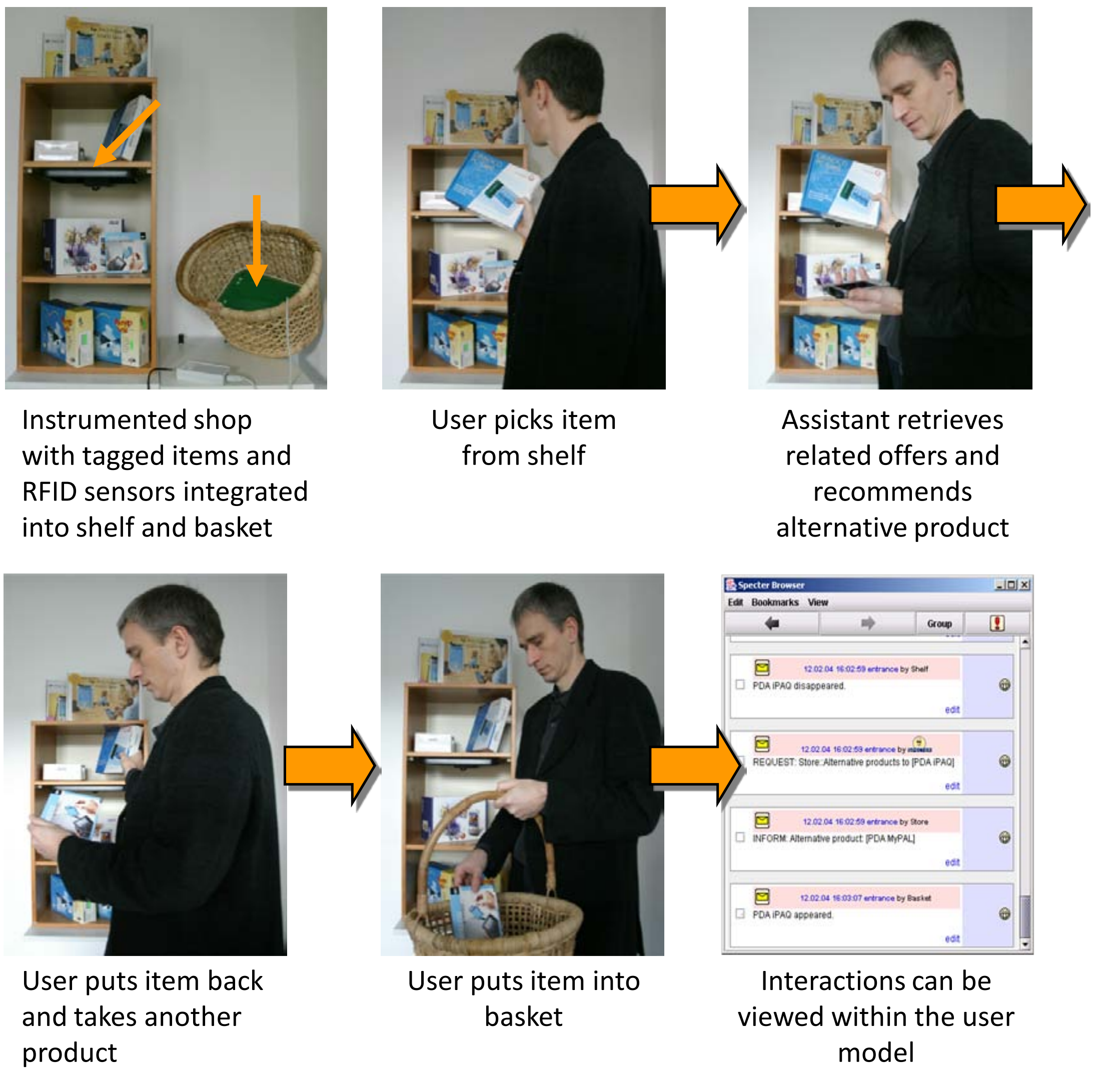
Now that we have seen how navigational activities are assisted in *REAL*, we will describe our assistance systems for the shopping activity. The *Smart Shopping Assistant* (SSA) is a prototype of a proactive shopping assistant, which offers value-added services in a shopping scenario (see also Schneider, 2003). In contrast to explicit human-computer-interaction, where the user controls an application's behavior through direct manipulation with buttons or dialogs of spoken command language, the *SSA* is one of our first applications that are based on the paradigm of implicit interaction between the user and the real environment.

During shopping, relevant user actions are for example to move around in the store, and to physically interact with products. Relevant context includes a user model, as provided by the aforementioned ubiquitous user model *UbisWorld*, the location of the user, and the products that are involved in a user's action as well as the overall product inventory of the store. The *SSA* comprises a central server, a modified shopping cart or basket, and instrumented shelves. Both, the shopping basket and the shelf, have been equipped with RFID readers as shown in the first image of Figure 57, to allow for the recognition of products tagged with RFID transponders. In addition to the basket, a PDA is used as the primary user interface. Alternatively to the basket, we have built a shopping cart with integrated RFID antenna and a tablet-PC mounted on the cart's handrail, hosting the shopping assistant, controlling the RFID reader, and connecting the cart with the central server via wireless network.

With this setup, user actions like taking a product from the shelf and putting it back can be recognized by repeatedly polling the transponders (IDs) of the products that are located in the antenna fields of each shelf board and the shopping cart. These sensor observations are fed as implicit input into the *SSA* application. For example, the second image of Figure 57 shows a user who picks up a product from the shelf to read the information on the package. The *SSA* application reacts to this action by retrieving related offers and product recommendations. Other value-added services include product comparisons and cross-selling recommendations. In addition, a plan recognition engine is able to recognize cooking recipes based on the current products in the cart, and suggest further ingredients. Another specialty of the *SSA* is its ability to apply a user model in order to adapt value-added services to the special preferences (e.g. nutrition habits) and needs of an individual customer.

Then the *SSA* sends its output to the presentation manager (PM) service, which is provided by the *SUPIE* infrastructure. The PM is responsible for selecting an appropriate display nearby to the user and for advising the display to open the presentation. In our example, the product recommendations are sent to the user’s mobile device, as shown in the third image of Figure 57. If the user would have used the instrumented shopping cart, the information could have been presented there. Alternatively, the information could have been shown on a public display next to the shelf instead of the comparably smaller display of the mobile device or cart.

The fourth image of our example shows how the user follows the advice of the system and puts the first product back onto the shelf. Then the user picks up the alternative product and drops it into the shopping basket. Back at home, the user can browse their user profile and review their interactions within the shop (see last image of Figure 57).


Instrumented shop with tagged items and RFID sensors integrated into shelf and basket
User picks item from shelf
Assistant retrieves related offers and recommends alternative product
User puts item back and takes another product
User puts item into basket
Interactions can be viewed within the user model
Specter Browser
Edit
Bookmarks
View
Group
edit

12.02.04 16:02:59 entrance by Shelf
PDA iPAQ disappeared.

12.02.04 16:02:59 entrance by [illegible]
REQUEST: Store::Alternative products to [PDA iPAQ]

12.02.04 16:02:59 entrance by Store
INFORM: Alternative product: [PDA MyPAL]

12.02.04 16:03:07 entrance by Basket
PDA iPAQ appeared.

Figure 57: User interactions with real products serve as input to the assistance system.

5.2.3 Multimodal Interaction with the Shelf

We have seen in the previous section how the *SSA* system responds to implicit interaction between the user and real products, namely i) to pick up a product from the shelf, and ii) to put it back. We have also implemented a multimodal user interface on the basis of a pocket PC device to allow for explicit interaction with the *SSA* system. In this context, we will refer to the interaction with the real products as extra-gestures, whereas intra-gestures refer to stylus input on the touch screen of the pocket PC. The user may decide to use *intra-gestures* (i.e. pointing via stylus to objects on the pocket PC's display) if the shop is too crowded or the products are locked away. In total, the *input modalities* include speech, handwriting, intra and extra gesture, while the *output modalities* include speech, text and graphics.

Our scenario assumes that the user possesses their own PDA and wirelessly connects it to the shelf device in the shop. Upon synchronization, the user can interact with a *virtual* set of products that is downloaded onto the pocket PC's display, with the *physical* set of products on the shelf, or with a *combination of physical and virtual products* making up the data space. Depending on background noise levels within the store, the user may also decide to interact with the products through the medium of *speech* using a headset. Figure 58 shows an example of an explicit product comparison query that combines speech with intra- and extra-gestures. If the store becomes very crowded, the user may switch to combined *handwriting-gesture* interaction.

Upon deciding on a product to purchase, the user can finally add the product to their shopping list and either continue to browse, or progress to the counter.

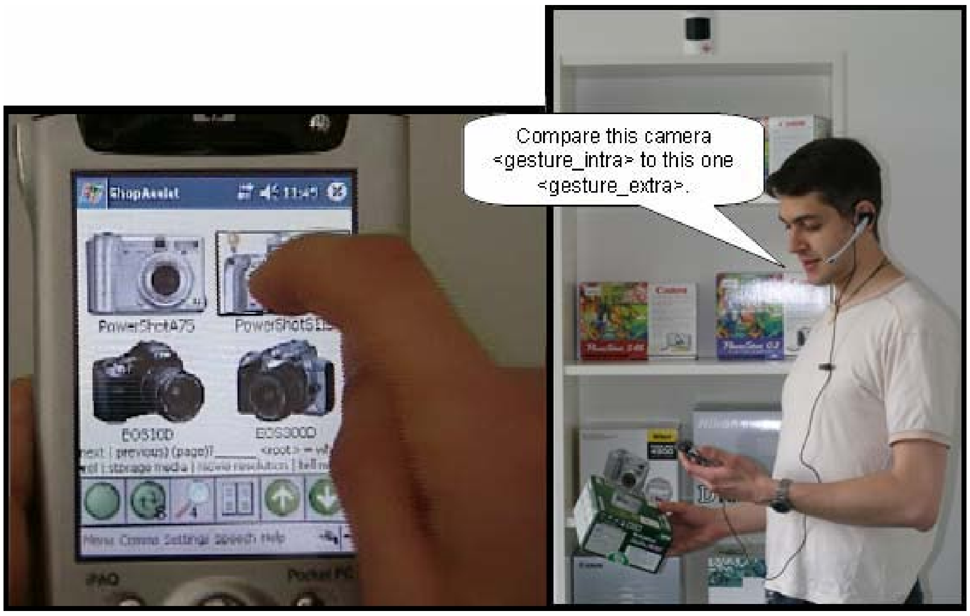

Compare this camera <gesture_intra> to this one <gesture_extra>.
ShopAssist
PowerShotA75
PowerShotS1
EOS10D
EOS300D

Figure 58: A combined virtual-physical interaction used to compare two products.

5.3 The SUPIE Architecture

Now that we have introduced the scenario and the user assistance applications of our intelligent environment, we will present its architecture and components. The components of the *SUPIE* architecture can be grouped into four layers, as shown in Figure 59, where assistance applications are considered as the top level. The applications have access to a knowledge representation layer, which includes location- and user-models. The service layer provides for indoor positioning and managing presentations on public displays. The bottom layer offers blackboard-style communication between the components. In the following we will briefly describe each of the layers, starting with the application layer.

5.3.1 Application Layer

The actual assistance applications of our intelligent environment use the the lower layers of knowledge representation and services to implement intelligent user interfaces. These assistance applications have already been introduced in the previous sections: the *BMW Personal Navigator* (BPN), the presentation agent *Cyberella*, the *SearchLight*, and the *Smart Shop Assist* (SSA).

All applications rely on the knowledge layer to adapt their interfaces to the individual user's profile. The BPN and *Cyberella* also adapt to the user's location which is determined by the *LORIOT* indoor positioning engine, as part of the service layer.

The *BPN* and *SSA* utilize the presentation manager (PM) service to serve navigational aid and product information to the user on shared public displays. There are several other applications known from literature that use public displays, such as the museum guide *PEACH* (Kruppa et al., 2003; Stock and Zancanaro, 2007), the posting service *PlasmaPoster* (Churchill et al., 2004), and the instant messaging service *IM Here* (Huang et al., 2004). By sharing access to display resources via the PM, applications like the aforementioned could also be integrated into the environment to extend the scope of user assistance.

Communication between the applications and the sensors of the instrumented environment, such as the RFID-augmented shelf and shopping cart, is realized by the communication layer that implements a blackboard for the exchange of event messages.

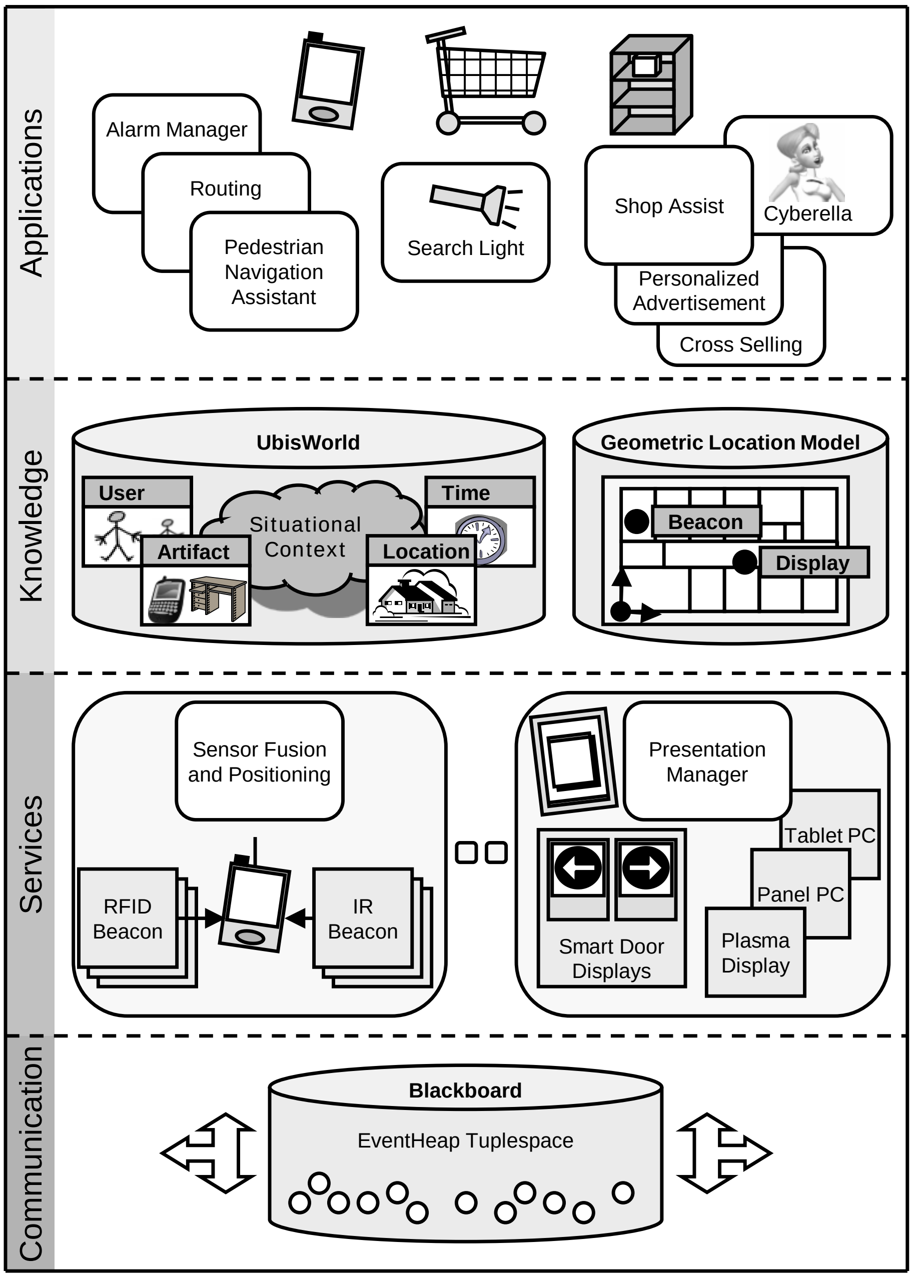


Figure 59: The software components of the SUPIE environment.

5.3.2 Knowledge Layer

Ubiquitous computing in an instrumented environment like *SUPIE* poses challenges and new applications to the fields of user modeling and location modeling. We use the *UbisWorld* ontology to represent *persons* by their user model as described in Section 4.1.4, and *places* through a symbolic location model as described in Section 2.4.4.2. The location model represents indoor and outdoor places at different levels of granularity, like cities, buildings and rooms, and serves as a spatial index to the situational context. It is possible to model various properties of spatial elements, such as room-temperature, noise-level, and humidity. Additionally, the presence of persons and physical objects can also be expressed.

In order to generate localized presentations and navigational aid, the positions of the user and ambient displays within the environment have to be precisely known. Therefore the symbolic world model is supplemented by a geometric location model for tasks like indoor positioning, map visualization, and route finding. It represents the building structure as regions, which are described by their coordinates and a symbolic identifier to reference their corresponding representation in *UbisWorld*. Navigational beacons, landmarks, and displays are also represented by the model in order to describe their visibility. The geometric model is created and maintained by the *YAMAMOTO* toolkit, which will be introduced in Chapter 7 of this thesis.

5.3.3 Service Layer

The service layer provides multiple applications at the same time with information about a user's position in the environment and offers access to the public presentation service. It hides the technical complexity of these services behind a simple interface, which is based on blackboard events.

For the **positioning service** we adopt a heterogeneous sensor approach, where a mobile terminal receives coordinates from infrared beacons as well as active RFID tags and estimates its own position using a dynamic Bayesian networks approach. The positioning service is presented later in Section 5.4.

Intelligent environments enable users to receive information from a variety of sources, i.e. from a range of displays embedded in those environments. From an assistance application's perspective, delivering presentations to users in such an environment is not a trivial task. While designing an application it is for example not exactly known which displays will be present in the specific presentation situation, and which of those displays might be locked by other applications. It is further unclear if other users are able to see the presentation, which could cause problems for the presentation of private information in a public space. Our solution addresses this problem by introducing the concept of a **presentation service** that provides an abstraction of the available displays. The presentation service provides a simple interface that allows applications to present Web content, such as HTML and Flash, on any display which is connected to the presentation manager by posting presentation requests on the blackboard. The presentation manager maintains a stack of presentations, and implements conflict resolution strategies to decide which content to present on which display. The presentation service currently resolves conflicts by considering the deadlines combined with priorities. It matches the display positions with the user's range, and presentations are queued until displays become available or multiple browser windows are opened (division in time space). The presentation service will be explained in more detail in Section 5.5.

5.3.4 Communication Layer

The communication and coordination within the *SUPIE* is based on the blackboard metaphor that has been first suggested for knowledge-based systems by (Newell, 1962). It describes a situation where several 'specialists' cooperatively solve a problem by communicating via a blackboard. The blackboard allows each component of a distributed system to contribute data, such as sensor readings, whilst other components can monitor the blackboard and process the data that appears. They can in turn write their results back to the blackboard. This asynchronous mode of communication is ideally suited for parallel computing, as (Velthuijsen, 1992) points out. More specific, we use a commonly accessible tuplespace (Gelernter and Carreiro, 1992), which provides an indirect interaction mechanism featuring the portability, heterogeneity, economy and flexibility needed for an intelligent environment. Processes post their information to the space as tuples (collections of ordered type-value fields) or read them from the space in either a destructive or non-destructive manner. In comparison to client-server designs, the anonymity and persistence of tuples offers a high degree of failure tolerance in dynamically changing environments (Johanson and Fox, 2002). If a requested service is temporarily unavailable due to a network failure or reboot, the stability of the client process will not suffer. We have chosen the EventHeap server and API, which have been developed at Stanford University as a framework for their iRoom (Fox et al., 2000) project. Similar implementations are available by Sun (Freeman, 1999) and IBM (Wyckoff, 1998).

In the next section we present how we model the user and parts of the real world in order to establish services and assistant applications supporting the users of our instrumented environment.

5.4 Indoor Positioning Service

Being at the right place at the right time is an essential precondition for any user interaction within the real world. Thus for any given task of a user, it is likely that a navigational sub-goal exists. Being aware of this, we have spent considerable effort in the development of indoor and outdoor pedestrian navigation systems, as published in (Butz et al., 2001; Baus et al., 2002; Krüger et al., 2004).

In order to realize the location-dependent output of route instructions, these systems need knowledge about the current whereabouts of the user. If the user stays outside of buildings, we can use a GPS (Global Satellite Positioning System) receiver for positioning. Unfortunately, the GPS signal can not be received in indoor environments, e.g. inside of a shopping mall or an airport terminal building. Therefore we have first implemented an indoor positioning infrastructure that is based on wall- or ceiling-mounted infrared (IR) beacons. Each IR beacon emits a 16 bit wide identification code once per second. Receiving such a signal with a mobile device, such as a PDA, gives a high probability that the user is standing in front of that particular beacon. Furthermore, since our location model represents the direction of the infrared light beam, we can also infer the user's orientation.

However, the disadvantage of IR beacons is that the PDA's IR sensor must be in the line of sight of the beacon and can thus be very easily blocked. To overcome this limitation, the

SUPIE environment implements a positioning service, called *LORIOT* (**L**ocation and **OR**ientation in **I**ndoor and **O**utdoor environmen**T**s), which combines infrared beacons (IR beacons) and additional active RFID tags to determine the user's position (Schwartz et al., 2005). *LORIOT* runs on a PDA and uses the built-in infrared receiver and an attached active RFID reader (PCMCIA card) as sensors. The RFID tags send their information via radio waves, which can be received even when the PDA resides in the user's pocket. But due to reflections and the damping of radio waves, receiving an RFID tag gives only little evidence that the user is standing in its vicinity and the signal misses directional information. By combining two sensor readings from IR and RFID with the help of dynamic Bayesian Networks, we achieve what we call an **Always Best Positioned** system (ABP system); as long as there are either IR beacons or RFID tags detectable, we will be able to estimate a position whose precision depends on the type of the sender. If both can be received, precision is even higher.

If the user is in between multiple beacon locations, we calculate the true location of the user by a weighted combination of their coordinates. Since the sensor readings are often noisy and unreliable, we have to apply filtering techniques to stabilize the position. Commonly Bayesian filters are implemented by Kalman-filtering, grid-based approaches, or particle filtering. We have implemented a new approach, which is efficient in time- and space-complexity and that provides a high scalability for in- and outdoor-positioning. Brandherm and Schwartz (2005) have introduced **geo-referenced dynamic Bayesian networks** (geoDBNs) to overcome the necessity for building a grid and therefore enable the calculation of a user's position on their own small hand-held device without a connection to an external server. Thus, position information is kept private and completely in the hand of the user. When a tag or beacon is discovered by the PDA, new geoDBNs are instantiated and associated with the induced coordinates. These induced coordinates depend on the stored coordinate and the type of the sender. Each network's belief represents the probability that the user is located at their respective position. The position of the user is then calculated by the weighted combination of the coordinates at which geoDBNs exist, according to each network's belief. To reduce calculation cost and memory usage, the number of instantiated geoDBNs is reduced by deleting the unused networks with lowest belief.

The resulting user coordinates are matched with the geometric location model, which is given by the knowledge representation layer, and result in a more meaningful symbolic location identifier. This symbolic name can be used to query the ubiquitous world model UbisWorld for the places' properties and spatial relations. Since the user is not tracked like in exocentric localization systems (see Section 4.1.5.2), it is up to the user's personal privacy considerations whether their position should be published in their user model or not. Even if it is published, the user can still restrict access to their position information to certain applications, such as the navigation assistant.

The information about the user's position within the environment can be used by the shopping assistant as well as by the navigation assistant. The prior starts the presentation agent each time a user enters the shopping room, and the position data helps to synchronize the product selection on the PDA with the actual shelf in front of the user. The navigation assistant can provide the user with customized graphical navigation instructions. These presentations are posted on the event heap as presentation requests to be scheduled by the presentation manager on available public displays in the environment. The details of the presentation service will be explained in the following section.

5.5 Managing Presentations for Multiple Users on Multiple Displays

Today, public displays are typically poster sized and non interactive, they present product information, advertisements or time-tables at fixed cycles. As the price trend for large plasma screens continues to drop, more and more public displays are being installed. Within the next years, polymer and organic LED displays of arbitrary size will enter the market, and they are likely to be integrated into all kinds of artifacts at negligible cost.

In anticipation of displays emerging everywhere, we investigate how to use them for the design of intelligent user interfaces, so that multiple users can benefit from multiple displays, which might be mobile, embedded or stationary, personal or public. The displays should convey a variety of information, originating from multiple applications, so instead of running a specific application on the public displays, we favor all-purpose World Wide Web technology such as HTML and Flash. In our *SUPIE* environment, the presentation service provides transparent access to all available display resources via a centralized presentation manager, which controls the presentation devices.

This approach allows for a non-exclusive usage of the displays by multiple applications. Besides product information and navigational aid, the same display might also provide messages and postings for instance, which are maintained by completely different applications. In such a scenario of multiple users and applications, conflicting presentation requests are likely to arise and have to be resolved. Therefore the presentation service has to implement intelligent strategies to allocate the most suitable display for a certain presentation and to divide the available displays by time and screen space, if necessary. The key to a personalized and localized presentation on public displays is spatial knowledge about the environment, i.e. the position of the displays and the user. Given that information, the presentation service is able to present individual content on a suitable display next to the user.

5.5.1.1 *Presentation Manager*

The Presentation Manager (Stahl et al., 2005) is an abstraction of the public/ambient display space from an applications point of view and provides a high-level interface to issue presentation requests (see Figure 60). In a public space with various displays we assume that a number of applications are running simultaneously and concurrently attempting to access display resources, so conflicts are likely to arise. Whereas canonical conflict resolution strategies could be “first come, first served” or priority based assignments of resources, we focus on rule-based planning; presentation strategies are modeled as a set of rules that are applied to a list of available displays and queued presentation requests. These rules generate plans at runtime, which define where and when presentations are to be shown. Applications post their presentation requests on the blackboard (implemented as an event heap). These requests include the mandatory and optional (marked with an *) arguments as shown in Table 3.

Table 3: The attributes of a presentation request event.

Event Key	Value
Source	**URL of the content**
Destination	**Display *or* location *or* user**
Type	**Image, text, video or mixed**
Expiration Deadline	**e.g. in 30 minutes from now**
(Minimum Display Time)*	**e.g. 60 seconds**
(Minimum Display Size)*	**Small, medium, large**
(Minimum Resolution)*	**e.g. 800x600**
(Audio Required)*	**Yes, No**
(Browser Input Required)*	**Yes, No**

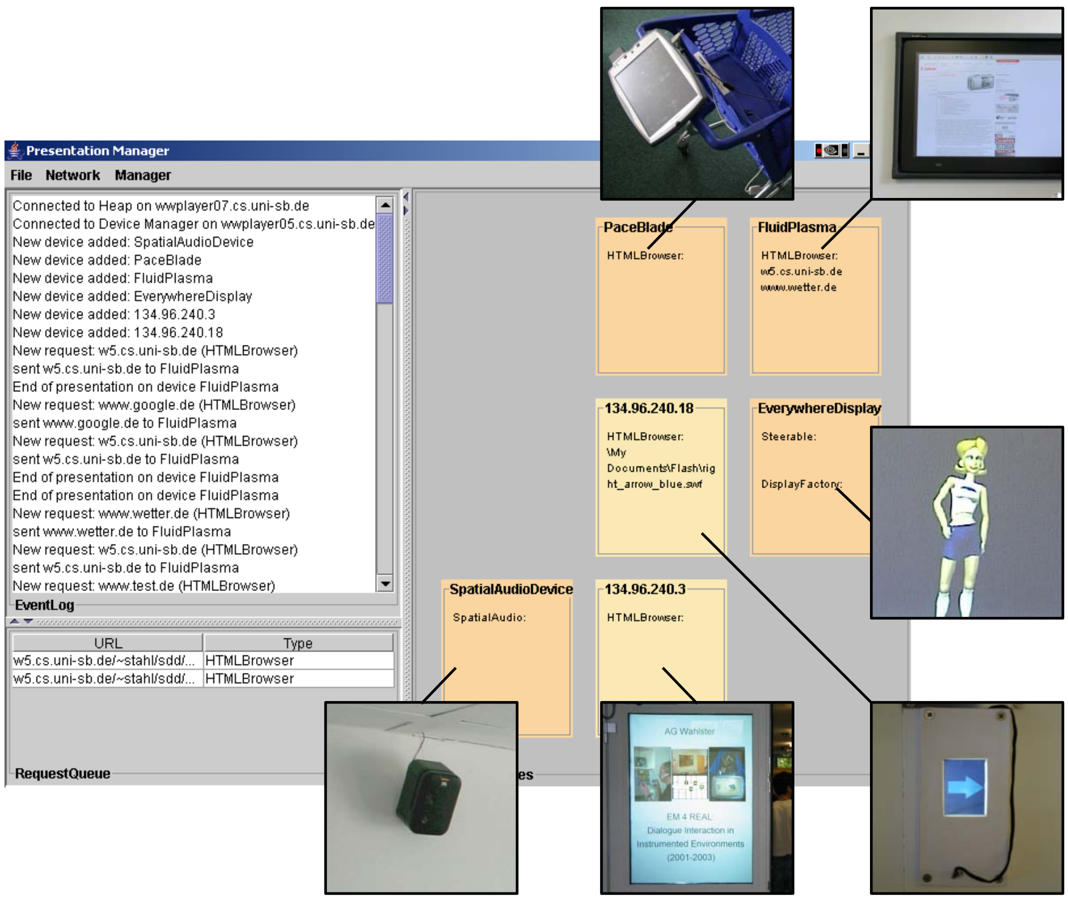
Based on the requests, the presentation manager plans the presentation schedule for all displays in a continuous loop. This loop is comprised of the following steps:

1. *Generate a list of feasible displays based on their properties and spatial proximity to the user.* This task is trivial if a particular display is given as the destination. If a location identifier (e.g. "office 1.06") is provided, *UbisWorld* is queried for local displays. If a person is the stated addressee of the presentation, first their location and orientation is retrieved from *UbisWorld* whereupon the spatial location model is queried to determine nearby displays, which could either be based on the Euclidean distance using coordinate vectors or - more reasonable - a qualitative distance, i.e. user and display reside in the same room. Having exact locations of users and displays, we are moreover able to examine the visibility of the presentation, and to consider the visual demands of its content in relation to the size, distance and viewing angle of the display. All displays fulfilling these requirements now constitute the list of feasible displays for a given user.
2. *Sort the list according to the following criteria (ordered precedence)*: idle state, minimum requirements (e.g. size), general quality and release time. The presentation will be shown on the first available display in the resulting list.
3. *Resolve conflicts by queuing requests (division by time) and splitting displays (division by screen space)*. If step 2 does not deliver any idle displays, we check if ending presentations will release a display before the deadline of the current request elapses. Displays for which this condition holds are sorted according to step 2 and the request will be queued on the first display (division of time). If still the request cannot be scheduled (the latter list is empty), we check if a display can be shared - preserving the given requirements (resolution, size) of the involved presentations (division of space). If this does not resolve the conflict, we attempt to re-plan currently playing presentations on different screens.
4. *Start over*. After applying these steps without success, we start over and consider more distant displays in combination with an audio notification, which is automatically added to the presentation by the presentation manager.

This set of rules provides coherent presentations in public spaces and resolves conflicts by dividing display resources in time and space; presentations are scheduled according to their deadline requirements and are delayed until resources are available (time). Screen space is shared if an appropriate device is available such that presentations are rendered simultaneously on the same screen in different windows (space).

Privacy issues require additional rules; each user can specify contents as private within the user model, for instance all navigational aid. The presentation service would then simply remove in the first step the displays that can be seen by other users.

Conflicts that arise from multiple users interacting concurrently can be handled by the same strategies. However, from the user's perspective, it is crucial to be aware of presentations intended for them and to avoid confusion caused by similar presentations for other users. Therefore, the presentation service notifies the user via an alarm signal (e.g. mobile phone vibration) on the personal device that is synchronized with the appearance of the presentation on a public display. If no such device is available, the content may be tagged by the presentation service with a personal graphical icon or sound that is stored in the user model.



```
Connected to Heap on wwplayer07.cs.uni-sb.de
Connected to Device Manager on wwplayer05.cs.uni-sb.de
New device added: SpatialAudioDevice
New device added: PaceBlade
New device added: FluidPlasma
New device added: EverywhereDisplay
New device added: 134.96.240.3
New device added: 134.96.240.18
New request: w5.cs.uni-sb.de (HTMLBrowser)
sent w5.cs.uni-sb.de to FluidPlasma
End of presentation on device FluidPlasma
New request: www.google.de (HTMLBrowser)
sent www.google.de to FluidPlasma
New request: w5.cs.uni-sb.de (HTMLBrowser)
sent w5.cs.uni-sb.de to FluidPlasma
End of presentation on device FluidPlasma
End of presentation on device FluidPlasma
New request: www.wetter.de (HTMLBrowser)
sent www.wetter.de to FluidPlasma
New request: w5.cs.uni-sb.de (HTMLBrowser)
sent w5.cs.uni-sb.de to FluidPlasma
New request: www.test.de (HTMLBrowser)
```

URL	Type
w5.cs.uni-sb.de/~stahl/sdd/...	HTMLBrowser
w5.cs.uni-sb.de/~stahl/sdd/...	HTMLBrowser

PaceBlade
HTMLBrowser:

FluidPlasma
HTMLBrowser:
w5.cs.uni-sb.de
www.wetter.de

134.96.240.18
HTMLBrowser:
\My Documents\Flash\right_arrow_blue.swf

EverywhereDisplay
Steerable:
DisplayFactory:

SpatialAudioDevice
SpatialAudio:

134.96.240.3
HTMLBrowser:

Figure 60: The Presentation Manager's window shows a log of presentation requests (left) and the registered displays (right). The insets around the GUI show fotos of these displays.

5.5.1.2 *Presentation Devices*

The motivation behind the presentation service is to make all displays in the *SUPIE* environment accessible for applications, such as public displays, projectors, PDAs, notebooks and desktop machines. Therefore we have implemented a small software called *ProMan* (Process Manager) that runs on each device (currently *Windows* and *WindowsCE* operating systems are supported) as a background task and provides the required registration and presentation functionality. The *ProMan* service periodically announces the device's presence to the presentation manager by posting "heartbeat" registration events to the event heap. These events include descriptive parameters, such as the network address, device type, screen size and resolution. The installation, usage, and configuration options of *ProMan* are described in Appendix C.

Additionally, the owners of personal devices may restrict the use of their displays to private content by providing their user-id. After a successful registration, the presentation manager can connect each display device via a TCP-IP socket connection in order to present the scheduled content. As a result the service provides a customized Web browser (Internet Explorer COM-Object), and it also supports the download and presentation of binary files, such as images or Powerpoint slides, by external helper applications. From the user's perspective, it is crucial to be aware of presentations intended for them and to avoid confusion caused by similar presentations for other users, which means to preserve consistency in the presentations given. Therefore, the presentation service notifies the user via a personal device by an alarm signal (e.g. mobile phone vibration), which is synchronized with the appearance of the presentation on a public display. If no such notification device is available, the presentation service can automatically disambiguate the content by adding a personal graphical icon or playing a sound that is stored in the user model.

5.6 Discussion

The objective of the *SUPIE* environment has been to create systems that proactively assist their users in solving their tasks. Whereas traditional applications provide a set of functions and leave the choice of the right tool to the user, proactive user assistance systems actively try to recognize the user's needs and goals in order to give useful suggestions. The system has to match the perceived actions of the user with all possible plans through plan recognition (see 4.2.3.1) so that it can inform the user about forgotten steps or suggest the next logical steps towards their goal.

In the scope of the project REAL we have designed two assistance systems, one for navigational tasks, and one that assists its user in their buying decisions. In contrast to traditional, function-oriented user interfaces, the user does not directly request help from the system through buttons or menus. Instead, the naturally occurring actions within the environment are sensed and interpreted as implicit interaction with the system. In the navigation task, the physical position of the user is sensed and matched with the route (which could be considered as a 'plan' that consists of a sequence of walking and turning actions). The system announces the next action and gives a warning if a wrong action is taken. In the shopping task, the system senses the user's interaction with real products as being pick-up and put-back gestures and proactively presents information about the products on a screen. The system also considers 'plans' for cooking recipes and suggests missing ingredients for possible dishes.

Finally, the project has certainly met its objective of demonstrating how the concept of implicit interaction can be applied to the design of situated user interfaces. However, the design process of the instrumented environment has been sometimes challenging. The design of graphical user interfaces (i.e. the desktop metaphor) has been a subject of research for more than twenty years, so that it is well understood today. In contrast, little is known about situated interaction with real objects in instrumented environments. During the course of our project, many decisions had to be made by intuition, and some design questions could not be answered at all.

One obvious question is: how to deal with multiple users in Instrumented Environments? Whereas our applications successfully assist a single user within our lab environment, any real-world scenario would obviously require support for multiple users. We had intentionally excluded this issue during the project *REAL* and postponed it to the follow-up project *BAIR* (*"Combining REAL and READY: User Adaptation in Instrumented Environments"*, 2005-2007, CRC 378, Funded by DFG), but we will briefly discuss the problems here. Considering the shelf, two or more users concurrently picking up products might confuse the system since the current RFID sensor technology is not sufficient to discriminate pick-up actions from multiple users. The current implementation interprets two consecutive pick-up gestures as a comparison action and would thus present inappropriate information for multiple users. Regarding navigational aid, we use ambient displays to present directions because we believe that they are a promising alternative to personal devices, where such an application would have to be implemented on a variety of platforms and operating systems. But if multiple users are in front of the same display, the screen will become cluttered with information, and privacy concerns also need to be solved. One possible solution would be to address users in groups (by their destination) rather than as individuals (by their name), or to let users queue in front of the display. The problem of designing user interfaces for multiple users and multiple devices has been addressed by the MU3I workshops series (Kray et al., 2004) and is an active research area.

5.7 Summary

In this chapter, we have reviewed the results of our research in the project *REAL*, which has been concerned with the question of how an instrumented environment can proactively assist its users. We have introduced our motivating scenario of a traveler, who intends to buy a digital camera in a duty-free shop at the airport. Our scenario assumes that the user is likely to require macro-navigational aid that helps him/her to locate an appropriate shop, and micro-navigational aid to assist him/her in finding a certain camera model on the shelf. In order to make an informed buying decision, the user also needs product recommendations and comparison information.

We have outlined how the situations of the scenario relate to the navigation and shopping assistance applications that we have actually developed in our instrumented environment called *SUPIE*. The focus of these applications is on implicit interaction based on sensor data, but they are complemented with multimodal user interfaces on a PDA that allow the user to formulate queries.

In this context, we have introduced the *BMW Personal Navigator* (BPN). In navigation mode, the *BPN* implicitly responds to changes in the position of the user and provides verbose directions via a headset and graphical instructions via wall-mounted ambient displays. We have further described the exploration mode, which allows the user to scroll a graphical map of the environment and to request details about landmark objects using a combination of speech and stylus gestures on the PDA's touch screen.

As a second system, we have described the *Smart Shop Assist* (SSA) application that proactively provides product information, related offers, and recommendations as the user picks up a product from the shelf. The shelf has been instrumented with RFID readers that identify products that are tagged with transponder chips. The proactive mode of operation is complemented with a multimodal interface that accepts combined input of spoken commands, intra-gestures with virtual products on the touch screen, extra-gestures with real products on the shelf, and handwriting.

In the second part of this chapter, we have described the layered architecture of our *SUPIE* environment, which comprises application-, knowledge-, service-, and communication-layers. The knowledge layer provides user and location models, which are realized by the *UbisWorld* ontology. We have also explained in detail how the service layer supports the designer of an application through the indoor-positioning service *LORIOT* and the presentation manager. Finally, we have described the event-based communication model.

We have closed the chapter with a discussion of the results and the open design question of how to support scenarios with multiple users.

6 The GOAL Design Method

In the previous chapters of this work, we have motivated the need for user assistance in everyday life situations, particularly in ambient assisted living scenarios. We have also pointed out that such systems must be easy to understand and use, otherwise they will not be accepted by the users. We have favored proactive user interfaces since they promise to reduce the user's cognitive load to operate them.

In Chapter 5, we presented actual examples that we have developed for navigation and shopping assistance. However, we made the experience that designing and implementing proactive user assistance systems in instrumented environments poses many challenges. In order to achieve the goal of developing helpful and usable systems, the designer requires a well-founded understanding of the user's needs, and deep knowledge about the capabilities of pervasive human-computer technology, such as RFID sensors and computer vision.

Shifting the focus from the desktop into everyday life requires a careful analysis of the users' needs with respect to the physical environment, i.e. the building structure and involved artifacts. Whereas the development of graphical user interfaces is well supported through integrated development environments (IDEs), we believe that current software engineering methods and tools are not sufficient to design instrumented environments, e.g. *UML* can't express spatial context.

In this chapter, we present our new design method for intelligent environments called *GOAL* (**G**eometry-**O**ntology-**A**ctivity Mode**L**), which is characterized thereby that it is based on a detailed three-dimensional model of the actual environment and an activity model that is derived from Activity Theory. Both models are linked together using an ontology. The method suggests a structured procedure to analyze the users' needs with respect to the surrounding environment in order to identify useful assistance features. Furthermore, the method supports the designer to make the necessary decisions for the instrumentation of the environment, particularly which sensors and actuators to use and where to place them.

We will outline how the additional effort to create a three-dimensional environment model is worthwhile as it pays off during the whole full cycle of system development from design to evaluation. It can even be used after the system's deployment for monitoring the state of the environment in real time and for remotely tracing breakdown situations in Dual Reality.

In the following chapters, we will introduce a toolkit that supports the *GOAL* method and show use cases where the method and tools have been successfully applied.

6.1 Motivation

As mentioned earlier in Chapter 4, ubiquitous and pervasive computing technologies bear great potential to assist their users in everyday life situations. However, as the number of devices and human-computer interfaces in our environment increases, the challenge is to significantly reduce the cognitive load that is required to learn and operate these systems. The perceived saving of time and attention has to be bigger than the necessary effort to learn and operate a new interface – otherwise it is not likely to be accepted by their users. We have seen in the previous Chapter 5 how proactive systems can be realized, which observe what the user is doing and match the recognized actions with a database of plans. Such agent-like systems implement methods from Artificial Intelligence to assist their users without explicit intervention. They provide necessary information, reduce cognitive load by externalizing tasks, remind of background tasks, warn to avoid failures, and assist in recovering from breakdown situations.

We have also seen at the end of the previous chapter that the design of such systems is ambitious, since it requires a precise model of the user's needs in order to implement useful features and to avoid undesired or disturbing system behavior. Furthermore, input sensors, output actuators, and suitable interaction modalities such as vision, gesture, and speech, have to be chosen, considering the environmental factors of the scenario. Finally, the hardware deployment in the environment has to be carefully planned. Especially geometric details such as viewing angles of cameras and displays have to be considered early in order to assess the hardware requirements. In summary, the most important design questions for intelligent environments are:

- Which user activities and tasks require assistance?
- How to map an activity model into interactions with artifacts?
- Which modalities should be used for system input and output?
- Which technologies should be used, and where should they be placed?

We believe that current software engineering methods and tools are not sufficient to develop such systems and that a holistic approach is required that considers the activity of the user within the physical environment as primary concept.

6.1.1 Activity Theory

We have chosen Activity Theory, which has been introduced before in Section 2.2, as the foundation for a new design method that helps the designer of intelligent environments to find answers to the questions above. Activity Theory allows us to model the complex relationships between the users and their environment in well-defined terms and principles. According to the theory, activity is defined as a triple of subject, artifact, and motive. Each activity can be hierarchically decomposed in actions and operations. Whereas Leontiev (1978) has only been interested in human activity, we assume that agent-like systems can also be conceived as subjects of their own which intentionally act on behalf of the user.

Our *GOAL* approach is partially inspired by the Object-Oriented Analysis method (OOA) (Coad and Yourdon, 1990) that preceded the well-known UML standard. The goal of OOA has been to define the requirements of software products based on the concepts of the Object-Oriented Programming (OOP) paradigm, which gained popularity in the early 90ies: class, instance, attribute, operation, and inheritance. Coad and Yourdon combined the concepts of OOP with semantic data modeling through *part-of* and *is-a* relations. Likewise, we combine the concepts of Leontiev's hierarchical activity model with an ontology that provides the semantic data model in order to define the requirements for user assistance systems in intelligent environments. Ontologies allow us to define concepts as classes and instances and to model *part-of* and *is-a* relations between them through roles.

One particularly important concept in Activity Theory is the artifact, which refers to objects that are required to perform an activity or action. If we look at activities, we also have to consider the related artifacts. Obviously, this definition includes tools, but also extends to other cultural artifacts, for example a television set. The most important difference between designing software for traditional desktop computing and intelligent environments is the strong relationship between the user's tasks and the environment. Whereas in the prior case the desktop computer is both tool and interface, activity in general relies on all kinds of artifacts that have evolved from the experience of generations. The idea of Ubiquitous Computing is to make artifacts part of human-computer interaction by augmenting them with sensors and output modalities - without affecting their normal usability in general.

6.1.2 On the Role of Time and Location

The relationship between activities and locations is particularly interesting for context-aware assistance and location-based services. If this relationship can be modeled, it can be applied by the system in both directions to improve interaction: i) if the current location of the user is known, the system can infer the most likely activities and adapt its user interface accordingly; ii) if the user's intentions are known, the system can provide navigational aid to guide the user to an appropriate place. The same is true for temporal context: i) activity recognition can be based on the time of the day and consider the duration between events; ii) if the scheduled time for an activity is known, the system can alert and remind the user. Ideally, an assistance system would consider the complex relationship between activity, time, and location.

As stated above, activity can be considered as a triple of subject, artifact, and object. Hence the possible location of an activity depends on the location of the required artifact. In case that the artifact is not ready at hand, the location of the artifact represents a destination, and the activity is likely to include navigational sub-goals and benefits from route directions.

The relationship between the activity and the artifact's location is depicted in Figure 61. Here it is interesting to consider that the mobility of the activity depends on the artifact's mobility; On the one hand, small mechanical tools like a hammer for example can be easily carried with the user, so they allow performing tasks in any place. For example, hanging up a picture can be considered as a mobile activity that can take place everywhere in the house. If, on the other hand, the artifact is not movable in a practical sense, the activity is bound to the location of the artifact. In the example of Figure 61, Activity 1 can only be performed at location A. The immobility can be either due to the size, weight, or infrastructure that is required to operate it; devices may depend on electricity, a data connection, gas/fuel, or water. Electricity can be relatively easy provided for small devices by batteries or radio transmission. Resource-demanding devices need direct access; a dishwasher for example needs a fixed waterpipe connection.

It is deduced that the activities that are performed at a certain location depend on the available artifacts and the arrangement of artifacts is planned according to cultural conventions. For example, at home, the kitchen comprises all necessary items for cooking-related activities, the bathroom is equipped for washing and body care, and the living room is fitted for home entertainment and relaxation.

This relation between space and artifacts is clearly quite important for the planning and design of instrumented environments, and should be part of the activity model.

Koile et al. (2003) represent the usage of places as **activity zones** – regions in which similar activity occurs. They describe a system that learns these zones from computer vision and suggest that such zones could be used to trigger assistive actions. The authors distinguish between two types of zones:

- Physical zones partition space according to the affordances of physical features – e.g. walls and furniture. Walls create a zone in which people might play music; furniture creates a zone in which people might read or talk.
- Activity zones partition space according to observed human activity – e.g. where people are sitting, walking, or standing.

Zones defined by physical form are useful as representations of location context, but ignore a person's actual use of space. Physical zones could be inferred from a static furniture configuration, but activity zones need to be learned from statistics of human behavior.

Besides the observed usage of activities in certain places, there are also social conventions that should be considered for design. For example, some people prefer to dine in the living room instead of the kitchen because the TV set is located there, which can cause some arguing in the family; eating in the living room is "bad manners". Watching TV in the kitchen on a portable device instead might be a solution. Harrison and Dourish (1996) have investigated the social aspects of location and distinguished between space and place:

> *"Physically, a place is a space which is invested with understandings of behavioral appropriateness, cultural expectations, and so forth. We are located in **space**, but we act in **place**. Furthermore, places are spaces that are valued. The distinction is rather like that between a house and a home; a house might keep out the wind and the rain, but a home is where we live."(Harrison and Dourish, 1996:69)*

To Harrison and Dourish the **sense of place** is a communally-held sense of appropriate behavior, and a context for engaging in and interpreting action. In Figure 61 we have symbolized this relationship between space and appropriate activities through the places X (activity 1) and Y (activity 2). Whereas the mobile activity can be performed anywhere and anytime, it may conflict with the sense of place X or Y.

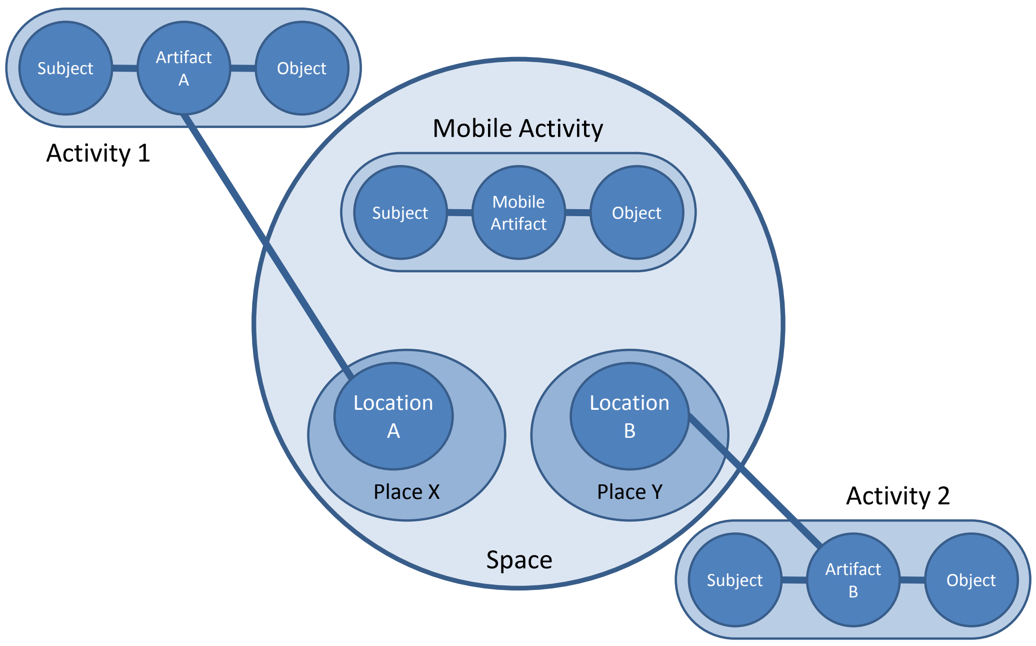


Figure 61: Relationship between activities, artifacts, place, and space.

Wikipedia defines the term sense of place a bit differently and more like the similar notion of the *spirit-of-place (genius loci)*:

> *“To some, it is a characteristic that some geographic places have and some do not, while to others it is a feeling or perception held by people (not by the place itself). It is often used in relation to those characteristics that make a place special or unique, as well as to those that foster a sense of authentic human attachment and belonging.”*

Nowadays, on the one hand, the increased mobility of artifacts, such as portable TV sets, provides convenience to our life. On the other hand, certain places are beginning to lose their “sense”. For example, 20 years ago we used to have a dedicated place in our home for having phone conversations with our friends and relatives. This place has been determined by the phone line and could not easily be changed. Likewise, we also used to talk to each other at certain times, e.g. Sunday evenings. Without spatio-temporal conventions like this, we would have missed our calls. With mobile phones, this relationship between activity and time and space is no longer relevant. Likewise, time-shift TV recording allows us to watch a show when we want it to see, not when it is broadcasted.

Aipperspach et al. (2008) critique that the increased mobility leads to “homogenization of space and time”, as traditional structures are diminished. Modern technology offers pervasive access to virtual environments throughout the home anytime, anywhere. This is mostly convenient, but also blurs the boundaries between work and home. Notebooks and PDAs provide access to work late at night, so for many the experience of “coming home from work” has become rare. This compromises the genius loci of our home environment and its restorative character, causing mental fatigue and busyness. Future trends in Ambient Intelligence foresee activity-adaptable rooms that can change their lighting and wallpaper design to different styles and moods; the same room can be used as office during the day and for a romantic candle light dinner at night, very similar to different scenes at a theatre stage. The question is whether this adaptability contributes to a sense of place and to the restorative character of our homes through heterogeneity, or further reduces it. If the switching between “places” in the real world also switches between virtual worlds, namely work-related and family-related content, it can help to reduce the blurring between both. No work-related documents would be accessible in the dinner setting, but pictures and music. However, designing such environments requires a detailed model about activity with respect to places in the real and virtual world.

6.1.3 Grounding the Design Process on an 3-D Environment Model

We conclude from the previous Section that the activities that one can perform in an environment are primarily connected to the required artifacts and their actual location. Moreover, behavioral appropriateness and cultural expectations also need to be considered for design.

Hence we suggest to base the design of a user assistance system on a careful analysis of activity and particularly location. We propose to begin the design process with the creation of a realistic model of the actual environment that reflects the geometry of the building structure as well as the artifacts inside, comprising furniture and appliances.

In the successive steps of the development process, the model visually supports the modeling of activities and their related actions and artifacts, and allows for linking them to the actual setting where the activity takes place.

Such a virtual environment model can also help to decide where sensors, actuators, and displays should be placed, and thus allows the designer to estimate how many items will be required and supports to justify the expenses for hardware. Regarding visual devices, like cameras and displays, the dependency between their position and field-of-view can be explored in the virtual environment and replace paper prototypes. Likewise, the range of antennas (for RFID or Bluetooth devices) can be estimated. However, in order to simulate how many access points a wireless network actually requires, sophisticated physical models would be required. Currently, wireless power transmission[47] seems to replace adaptors for mobile devices in the near future; since the transmission range between coils is limited, their placement matters. Besides the general trend to wireless technologies, wires still have their advantages of high bandwidth and reliability. Hence a 3-D model could be applied to plan the wiring of instrumented environments, e.g. with bus systems for home automation.

Developing user assistance systems (particularly for elderly users in the AAL domain) requires a user-centered design approach that includes the evaluation of new systems in an early stage in order to validate their acceptance by the target users. If a 3-D model of the environment and its instrumentation exists, it can be used to demonstrate the setting in Virtual Reality before expensive hardware needs to be acquired.

The same model can also be applied during the implementation phase of a system; software components can be connected to virtual sensors and actuators to test whether they comply with their specification even before the environment is set up and the components are deployed. Furthermore, the concept of Dual Reality (2R), which has already been introduced in Section 1.4, is characterized by a tight coupling between the real world and the virtual 3-D

[47] http://www.wirelesspowerconsortium.com/

model. Such a 2R model allows the developer to visualize the current state of the real environment in VR, and even to manipulate it through the avatar and virtual sensors. We will explain later in this Chapter how 2R can be applied to monitor the operation of instrumented environments and how to incorporate user feedback for debugging the environment.

Furthermore, the environment model can be used for documentation purposes after deployment. Once the hardware has been installed in the environment, some devices like RFID readers and antennas will be invisibly mounted, so that the model is useful to find them if maintenance is required. Replacing empty batteries is another issue that can be eased by the model.

Finally, if a model of the existing infrastructure is available, it can support the refitting or upgrading of an existing environment with new devices. Scalability is an important aspect for AAL, since it is difficult or even impossible to predict when certain assistance systems (such as cognitive prosthetics) will become required by a user since it depends on their individual ageing process. The model can explain where necessary plugs are available and ease the setup and configuration process of new devices, which also comprises issues of end-user programming. An environment that is adequately prepared to support the plugging of new devices through such a model could be considered as *AAL-Ready*.

In order to support the documentation and maintenance of the environment's sensors and actuators, we propose to complement the entities of the geometric model by a dictionary of unique, machine-readable identifiers through the use of ontologies. This implies a mechanism to define consistent symbolic names for places and artifacts that can be used throughout the whole development process from specification to documentation. The symbolic names can also be used to link further details to the model. Technical information for example is of interest if a device fails and needs to be repaired or replaced. Typical device information would be the exact model and vendor, price, and warranty expiration date. In case of visual devices their resolution does matter, and in case of radio devices the transmitting frequency may be important.

The *SemProM* project[48] describes such a maintenance scenario, where objects are linked with a **digital product memory** (Wahlster, 2007) that keeps a diary of important information throughout the whole product life cycle, from its production to where and when the object has been bought and how it has been used since then. A digital product memory may combine information stored at the object itself with external sources, such as the Object Memory Server (OMS) proposed by (Schneider et al., 2008). The OMS provides links to any kind of information about the object, ranging from user manuals and technical specifications to sensor data associated with the object.

6.1.4 Design in the Scope of an Iterative Development Process

We have seen in Section 2.1.4 how an iterative interaction design process can be applied in order to improve the usability of interactive systems through the evaluation of prototypes before the final product is being implemented. Initial assumptions about the users' needs are being refined during each cycle of the process, until the requirements are stable and the design meets the usability goals. Our proposed *GOAL* design method can be embedded for

[48] http://www.semprom.org/

example in Boehm's (1988) iterative spiral model of the system development process:

1. **The new system requirements are defined** in as much detail as possible by assessing the users' needs.
2. **A preliminary design is created** for the new system:
    a) A location model is created that represents the geometry of the building, its rooms, and artifacts;
    b) The current usage patterns are mapped to an activity model;
    c) Interaction design concerns the choice of modalities, sensors, and actuators. They are added to the location model and refer to the activity model.
3. **A first prototype of the new system is constructed** from the preliminary design.
4. **The prototype is evaluated** in terms of its strengths, weaknesses, and risks; either by focus groups, demonstration facilities, or in a VR/2R environment;
5. **A second prototype is evolved** by repeating steps 1 to 4:
    a) Refining the requirements for the second prototype based on the findings from the evaluation, i.e. improving its usability;
    b) Planning and designing the second prototype;
    c) Constructing and testing the second prototype; in this phase, a fully implemented system could be tested in a living-lab experience.

6.2 Capturing the User's Needs

User assistance technology should not be an end in itself, but focus on real deficiencies. It is of critical importance for the success of Ambient Intelligence projects to assess and capture the actual needs of the target group, so that the system developers can put themselves in the position of the user. Especially in the domain of Ambient Assisted Living it is difficult for the user interface designer to anticipate the limited (sensory, motor, and cognitive) abilities of elderly users and their living environment. In the following, we will give a survey of current elicitation techniques that support a user-centered design process and discuss their advantages and weaknesses. Finally, we will relate our proposed activity-based design method to the presented techniques.

6.2.1 Overview of Elicitation Techniques for AAL

It is advisable to begin a project with a **literature review** on the domain. It is often neglected, leading to analysis effort without new results. It is also recommended to consult **professional knowledge**; even if a user-centered design approach is intended, some information has to be asked from the professional that have this knowledge.

The organization of a **workshop** is usually the first step in a project, where all stakeholders are invited in order to meet the technical partners and to discuss the needs of the user group and the technical possibilities.

In order to overcome the limited domain and technical knowledge of the workshop's participants, Sponselee et al. (2008) suggest **role plays** to get acquainted with the scenario, technology and processes. It leads to natural reactions on usability and acceptance. The results can be improved if participants are videotaped first and then analyzed with the actors (nested video).

A **virtual persona**, according to Cooper's (1999) definition, is a construct of fictitious, specific, concrete representations of target users. It is the archetype of a user which will help us to design characteristics, navigation, interactions and even the visual design of the product. A Virtual Persona is not a list of tasks, but a narration describing the daily life of a user type. Virtual Personas are developed for the technical partners of a project in order to share an accurate and consistent idea of the target user group. Buiza et al. (2008) describe how they applied Virtual Personas to support a User-Centered Design approach to system design in the *i2HOME* project. The Personas helped the developers to focus on the goal user, making the devices usable and intuitive and adapting the systems to their skill and expertise.

The technique of leaving a **cultural probe** has been introduced by Gaver et al. (1999) in order to get in contact with elderly user groups, bridging the generational gap. Gavers group focused on a new understanding of technology and aesthetics, so they handed out materials to the seniors, including postcards with questions, a disposable photo camera, and a diary, to be returned with information about their local culture and living environments. The probes triggered the people to document their lives in a more playful and expressive way than questionnaires could. Cultural probes are often used today for ethnographic research and are a departure point for qualitative interviews.

Ethnographic research on current user practices can be supplemented by **Proxy-Technology Assessment (PTA)** (Pierson et al., 2006). The users receive existing technical tools which provide similar functionality and usage than the technology that is to be developed. Based on their experience with this proxy technology, the user group is able to foresee how they could benefit from the newly developed system in the future, e.g. through video surveillance in home care (Veys, 2008).

Initial **interviews** with goal users can primarily give insight into their attitude towards technology, which includes their expectations but also concerns and fears. The individual interviews can be completed in a more collective way with **focus groups**. They are useful for confirmatory or disconfirmatory information obtained with other techniques. Both techniques are discussed in the scope of AAL projects by Subasi (2008).

The techniques described above are suitable to assess the users' needs in the first phase of a project, which is usually referred to as requirements analysis. In the following design phase, these findings are the basis for decisions about the system architecture and user interface. In a user-centred design approach it is typical to implement a working prototype in the third phase which can be presented to and evaluated by the goal users. In a cyclic (or iterative) development process, such as Boehm's spiral model, this is the beginning of a second cycle that refines the requirements.

As soon as at least a mock-up prototype of the system has been built, it can be shown to Focus Groups again in order to get first feedback of the users' feelings about it.

If the prototype is already working, its usability can be evaluated in a **demonstration facility** under lab conditions. De Ruyter et al. (2005) suggest that technology should be tested in a realistic setting like their HomeLab which emphasizes on a home-like character so that users can imagine themselves in their home situation. Recently, specialized **living labs** have been built for this purpose within research institutes that closely resemble apartments; among others these are the Bremen Ambient Assisted Living Lab (BAALL) at DFKI Bremen, the AAL Environment of the Fraunhofer IESE, or the intelligent dormitory iSpace of the intelligent inhabited environments group at Essex University; see Section 3.1 for an overview.

For long-term studies beyond a lab demonstration, **in-home experience** means that a fully functional demonstrator is deployed in the living environment of a target user. The functionality should be real and working in order to demonstrate the technology as well as the care service processes. Otherwise, users can not experience the usefulness, resulting in an inaccurate analysis. It is notably that users will not accept technology in their homes if it has unsolved physical issues like disturbing cables, led lights, and fan noise. If people do not accept the system, they will not use it even when they actually need it (Sponselee, 2008).

The latter evaluation techniques are work-intensive and expensive, but also quite effective. However, one must consider that the findings are person- and situation- specific; it is therefore important to carefully select representative individuals for testing.

The techniques described above are briefly summarized in Table 4, which also includes comments on lessons learned and possibilities for AAL according to the results of the AmI2008 Workshop on Capturing Ambient Assisted Living Needs (Geven et al., 2008).

6.2.2 Bridging the Gap between Users' Needs and a Formal Specification

From a critical point of view, one can argue that the elicitation techniques above only give rather subjective, if not anecdotal results: subject A said this, subject B said that. It is difficult to generalize and formalize casual statements made by individuals.

During the following design phase, but at least when entering the implementation phase of a user-centered development cycle, the individual statements need to be transformed into a system specification that formally defines the properties of the product in a way that it can be validated against by the developer and the client. From a requirements engineering point of view it would also be important to document the roots of the statements and requirements, especially those of the stakeholders that are relevant for the contracts to be made, and to make them traceable in case of a dispute.

There is currently a large gap to fill for the designer between the anecdotal user statements and a formal specification for the system engineers. This is exactly where our activity-based design approach is targeted to. We propose an intermediate step where the informal, verbose statements are transformed into a tabular representation of activities and actions, with references to the user's needs and the situational context. We argue that such a table, if aided with a design tool, serves as a better rationale for design decisions including the instrumentation of the environment with ubiquitous computing technology.

Of course, focusing on current practices in such a way can only support user-centered design processes that aim towards improvements in usability and user experience, whereas disruptive innovations typically benefit less from such knowledge. These issues will be further discussed in Section 6.3.4.1.

Table 4: Overview of Elicitation Techniques. Source: Results of the AmI 2008 Workshop on Capturing Ambient Assisted Living Needs (Geven et al., 2008)

Techniques	**Lessons learned**	**Possibilities for AAL**
Literature Review	Needs to be in a context-related form Often neglected, leading to analysis effort without new results	Usage Context
Professional Knowledge	Knowledge of gerontologists and medical people Can't be replaced by interviewing only users themselves	Expert analysis and advice is important and must be added to previous domain knowledge
Workshops with all stakeholders	Not suitable for needs analysis or usability tests No real project group People being unfamiliar with technical possibilities	Preliminary domain analysis Complete picture of the outcomes of the literature review, heuristics
Cultural Probes	Avoid childish things for elderly Topic is important Kit may be used to present results, but meaningless	Learn about patterns and values Preparation for interviews
Ethnographic Approach	Expensive, high effort Multiuser approach shortens time Better than questionaires	Good for contextual information After 7 days, real patterns
Proxy Technology Assessment (PTA)		Use technology that is already available on the market To implement into personas
Interviews		Best results when conducted in a comfortable place, e.g. at home
Focus Groups	Elderly have difficulty to speak Opinion marker	Give confirmatory information about users' needs
Virtual Personas	Help team to imagine the target user group No new knowledge	Useful to communicate user needs to technical partners
Role Plays	Actors as care giver Vivid representation of future	Nested video instead?
Activity-Based Design Tool		Visualization of feedback patterns of usage and activity
Demonstration Facilities	Technology has to be implemented and reliable, Lab should exude a home-like character Expensive, but close to realitv	PTA could be cheaper alternative
In-home Experience	Most intensive and expensive Best results	

6.3 The GOAL Design Method

The design process begins after the users' needs have been assessed and documented through the elicitation techniques that we described in the previous section. These informal findings need to be structured and transformed into a system design that describes its scope, functionality, and user interface. The design needs to be further formalized in a system specification that is suitable to be passed on to a team of software developers. Particularly in case of intelligent environments, electrical engineers and craftsman also require specifications how to install wiring, lighting, displays, and other sensors and actuators into the walls, ceilings, and furnishing of the room.

Metaphorically speaking, our proposed design method *GOAL* (**G**eometry-**O**ntology-**A**ctivity Mode**L**) (Stahl et al., 2008; Stahl, 2009) is based on three pillars, as indicated in Figure 62, namely the geometric environment model, the activity model, and the ontology, which provides the symbolic names that tie both models together. As argued earlier in the motivation, we ground our design method on a detailed, geometric location model of the actual environment where the assistance system is about to be installed. All elements of the geometric model shall have their symbolic counterparts in the ontology. The activity model shall define activities that reflect the users' needs and contains references to related artifacts and their locations in the environment model.

Figure 62: The anatomy of the GOAL (Geometry-Ontology-Activity ModeL) Design Method.

These pillars yield the interaction model that describes how the users will be assisted in their activities by the various components of the instrumented environment, with focus on interaction modalities, sensor input, and actuator output.

The system design will finally be derived from the interaction model. On the one hand, it can serve as system specification for the implementation phase. The geometric model can be used for indoor-positioning and location-awareness, and the ontology can be used to identify parts of the domain and system for messaging between components.

On the other hand, the interaction model provides the necessary terms and illustrations for the system documentation, which comprises manuals and help systems for the end user and technical documents for maintenance purposes.

As already mentioned in Section 6.1.4, the resulting design should be presented and evaluated by its target users early in the design process. The geometric location model can provide a virtual testing environment before the actual instrumentation is installed. Once the system is deployed, a Dual Reality model can empower the developer to monitor and manipulate the real environment remotely in VR. This technique is particularly useful for remote debugging and incorporating user feedback in case of malfunctions.

The *GOAL* design method can be structured into the following sequential steps, which will be explained in detail in the next sections of this chapter:

1. **Modeling the environment** with respect to its physical shape.
    a. Modeling the building structure, including walls, doors, and windows;
    b. Adding the furnishing, in particular storage elements like shelves and cabinets;
    c. Inserting the relevant artifacts and appliances.
2. **Extending the ontology** with unique symbolic names for the elements of the geometric model.
3. **Activity Modeling**. Create a situational semantic activity model that reflects the users' needs.
4. **Interaction Design**.
    a. Analyze and utilize existing human-machine interaction;
    b. Choose sensor input and actuator output technology;
    c. Place input and output devices into the environment model.
5. **Implementation** of the system. The models can be used for location awareness, indoor positioning, and activity recognition
6. **Simulation and Evaluation** of the system design in Virtual Reality (VR) to let target users experience the environment prior to its actual installation.
7. **Monitoring and manipulating** the real environment in Dual Reality (2R).

6.3.1 Modeling the Environment

The easiest way to get started with the design of an intelligent environment would be to make notes on a floor plan or paper map, showing the environment as a top-down projection in two-dimensional space. While this representation gives a good overview, it is not possible to represent stacked objects, e.g. in a kitchen, or overlapping regions, e.g. staircases. It requires additional sketches to depict the vertical dimension. Furthermore, it is difficult for average people to mentally visualize a situation in 3-D if only a map is given.

Three-dimensional representations are well known today from computer graphics, and a wide variety of tools and languages exist to create and render virtual worlds in high quality even on low-end computer hardware. For a realistic display, surface attributes can be defined that describe the appearance in terms of color and glossiness. It is also typical practice to map image textures on surfaces.

Such a representation is well suited to model the environment with respect to its physical shape. We suggest beginning the modeling process with the building structure, including walls, doors, and windows. The room layout can be geometrically represented by spatial regions that are defined over vertices (coordinates in three-dimensional space) and edges. Next, the furnishing can be added to the rooms. Elements like shelves, cabinets, and tables can be represented as polygon meshes. Finally, artifacts like a vase or appliances like a dishwasher can be modeled accordingly and placed in or on the furniture.

However, by doing so with general-purpose tools for 3-D visualization, like SketchUp (3.4.1) we are ending up with a pile of polygons that are optimized for visual rendering, but meaningless in other respects. In order to apply algorithms that derive higher-level information, such as route knowledge, we need to annotate the polygons with some semantic knowledge, as described in 2.4.1. This can be achieved through the usage of ontologies, as we will see in the next section.

Another interesting aspect is the simulation of physical actions within the virtual world, e.g. the effect of motor actuators or the motion of robotic agents. Therefore we need extensions that allow us to define the physical properties of the modeled objects, such as their weight, and their physical assembly through joints or hinges that restrict their degrees of freedom.

Finally, for navigation purposes, multiple levels of a building need to be modeled and the coordinate conversion between model, metric and geographic units (Longitude and Latitude) should be supported by multiple frames of reference (see also Section 2.4.2).

6.3.2 Using Ontologies to Create a Symbolic Location Model

For the second step we suggest to create a symbolic representation of the domain, which defines a unique symbolic identifier for each **artifact** and **spatial region** of the environment model, and also subjects and roles. The goal of this step is twofold: On the one hand, these identifiers shall provide the link between the geometric model and the activity model that is to be created in the third step. On the other hand, in conjunction with literal names, these symbols are particularly suitable to guarantee that a consistent notation will be used throughout the whole development process in the specification and documentation materials. Furthermore, the symbols shall be used for the actual implementation of the system; they afford the realization of seamless user interfaces by enabling the ubiquitous exchange of contextual information between different assistance components.

Typically, ontologies represent taxonomies of concepts that can be used to classify part of the existing things in the world, and are organized in branches. Besides, ontologies can also be used to list concrete instances. Today, ontologies are rarely constructed from scratch, but rather existing frameworks are extended with new concepts and instances, as needed.

The *UbisWorld* by Heckmann (2006) (see Section 2.4.4.2), for instance, comprises several specific ontologies, such as the *General User Modeling Ontology* (GUMO), the *UbisEarth* ontology of geographic and local places, and the *UbisOntology* that describes physical things in the world, including persons and artifacts.

The *UbisEarth* spatial ontology already includes a complete listing of geographic names for continents, countries, regions, and cities (see Figure 15); street names are partially available for certain cities. If a new environment has to be modeled, instances mainly need to be created for the building and its rooms and spatial regions, respectively.

Tagging spatial regions of the geometric location model with the symbolic location identifiers of *UbisEarth* results in a hybrid location model (see Section 0) that combines the advantages of Cartesian coordinates with spatial relations, like *part-of* or *inclusion*.

Furthermore, the physical ontology needs to be extended with instances of the room furnishing and artifacts. The *UbisOntology* for instance already provides an extensive list of elements for the AAL domain, such as household devices like kitchen appliances. New concepts can easily be added online through the Web-based *UbisEditor* (Heckmann et al., 2009; Loskyll et al., 2009), if needed. The editor also supports multilingual literal names for each concept, so that the localization and translation of applications and their documentation is well supported. CODA (**C**ollaborative **O**ntology **D**evelopment **A**pplication) (Loskyll, 2009) supports the collaborative development of ontologies through an interface structure that integrates the *UbisEditor* with a search function and map view.

6.3.3 Activity Modeling

In the third step of the proposed design method, the users' activities and needs shall be modeled according to Leontiev's Activity Theory (see Section 2.2). The goal of this step is to organize and tabulate the informal, sometimes even anecdotal, results gathered by the elicitation techniques of the first phase.

Activity Theory is characterized by its hierarchical structure where activities are decomposed into actions, which in turn are decomposed in operations. Activities are motivated by (more or less abstract) needs, whereas actions are pursued towards the achievement of concrete goals. On the lowest level, operations establish the necessary preconditions. The distinction between actions and operations is made by the cognitive effort that is required for the subject to perform them; operations can be carried out automatically without conscious thinking about how to do it, whereas actions typically require some planning beforehand. Of course, the separation between activity, action, and operation also depends to a certain degree on the interest of the modeler and the scenario that is to be analyzed and described.

For our purpose of designing user assistance in instrumented environments, it is important to know the subject who performs an activity, the artifacts that are involved, and when and where an activity is typically performed. Figure 63 illustrates some activities of daily living in the morning from 6:00 to 9:00 am, including the preparation of breakfast. The image sequence on the left hand side symbolizes when and where the different activities are performed. On the right side the according appliances are shown, which represent some of the artifacts that are involved in the activities. For example, between 6 and 7 am the man shaves in the bathroom using an electronic shaver. Meanwhile, the woman prepares the breakfast in the kitchen using multiple appliances like a toaster, microwave, cooking stove, and refrigerator. In order to describe such a scenario with Activity Theory, we suggest extending the (subject, artifact, object) triple to a five-tuple that also describes the spatial- and temporal (situational) context of the activity. Furthermore, we propose to include references to virtual artifacts and human-machine interaction, which will be introduced in more detail in the following section on interaction design. We will now motivate each category before we formalize the model.

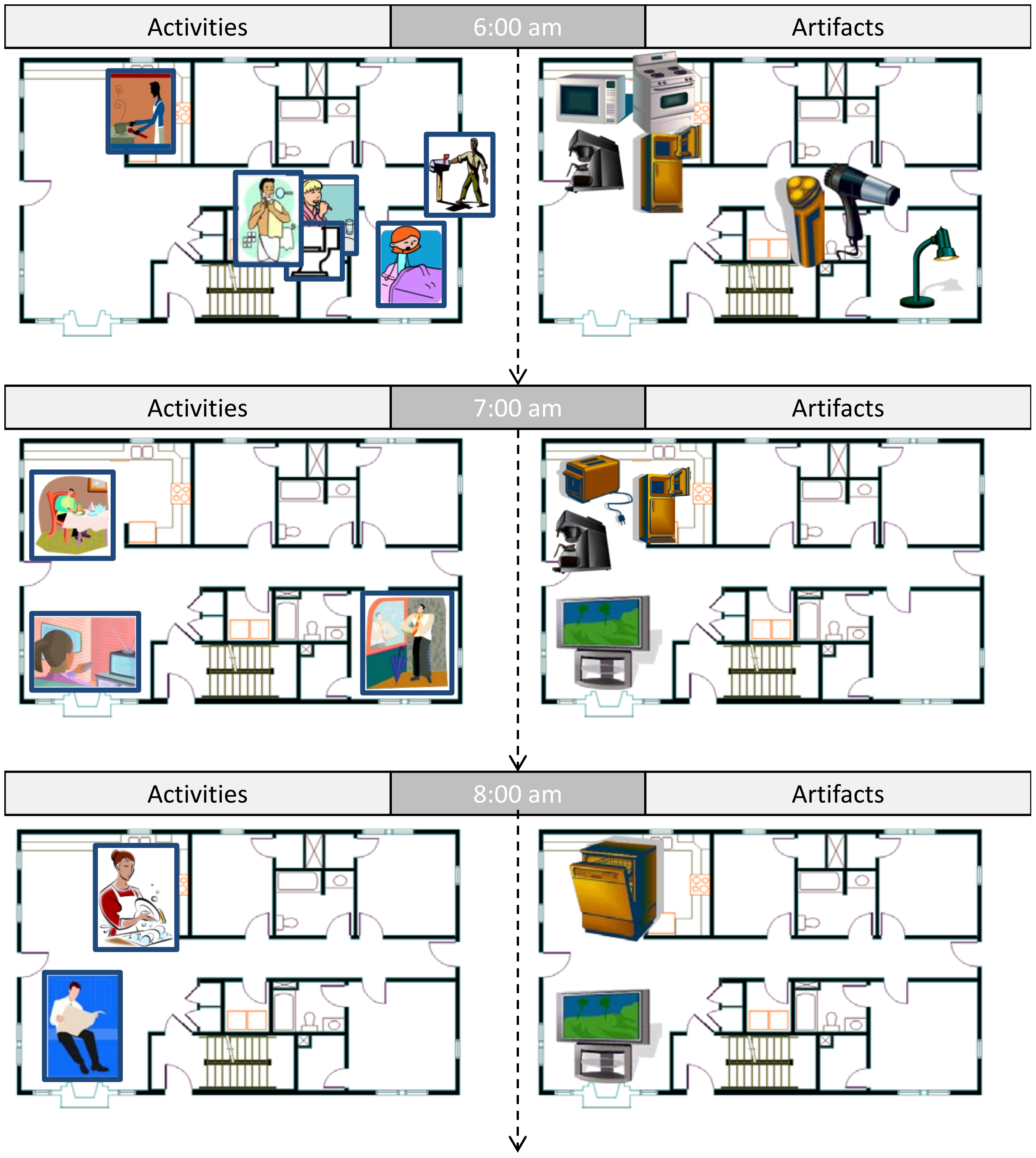


Figure 63: Activities and artifacts that are used at home in the morning. Source: I. Kobayashi.

6.3.3.1 *Subject*

The subject of the activity (or action/operation) refers to the actor that performs it. The actor shall either be an individual person, or, more generally speaking, a role like "*user*", "*care giver*", or "*care taker*". Furthermore, categories related to age ("*child*", "*adult*") or gender ("*male*", "*female*") can be assigned to activities, as necessary. For example, a certain activity like playing games may be typical for children, whereas interaction with potentially dangerous artifacts like a cooking stove is not suitable for them. Besides roles, Virtual Personas could be used that are known from requirements engineering and represent stereotypical users of a system in high detail, as described in Section 6.2.

Vice versa, the various components of a user assistance system itself can also be conceived as a subject in order to represent its assistive actions according to the terms of Activity Theory. This allows representing interaction between the environment and its users in a symmetric and consistent way.

The required persons, roles, and systems shall be modeled in the ontology, as described above, so that their symbols are available to be referred by the activity model.

6.3.3.2 *Physical and Virtual Artifacts*

The notion of the artifact as a mediator between subject and object has already been introduced in Section 2.2.2, and we assume that all necessary artifacts shall be modeled in the geometric environment model as well as physical elements in the ontology.

It is noteworthy to consider how the World Wide Web (WWW) adds another dimension to artifacts, namely according to Pederson (2003) the "physical – virtual" perspective: Physical artifacts often have virtual counterparts on the Web. For example, most products have digital representations, including textual descriptions, technical specifications, photo images, and software, such as firmware updates and drivers. The same is true for user manuals, which are often (and sometimes only) supplied in electronic form.

Electronic devices that support the Universal Remote Console (URC) framework (Alexandersson et al., 2009) which serves as an interlingua for other protocols, such as the well-known UPnP[49] standard, expose their functionality to the Web. In addition to their physical user interface, they also provide a virtual user interface that allows to remote-control such devices through a universal remote console according to ISO/IEC 24752 (2006) or user assistance system on a task-level. Ultimately, the Internet of Things (Floerkemeier et al., 2008) describes a scenario where each artifact is somehow connected to the internet via wireless communication technology, e.g. RFID, and can transmit information about its life cycle, from production to transport, as well as current or recorded environmental conditions. Dual Reality scenarios imply a 1:1 relation between physical and virtual artifacts.

Whereas in our activity model the physical artifacts are represented by concepts that are provided by an ontology, their virtual counterparts (documentation and interfaces) may be directly referred to by their URL or URI given that they are connected to the WWW.

6.3.3.3 *Objective*

The objective (motive) of an activity shall be represented in the ontology and supplemented with a verbose description, and optionally include a reference to the prior assessment of the users' needs to trace their roots. The same applies to the goals of an action and the preconditions of an operation. In case that the subject is the assistance system itself, the needs may refer to the intended user experience (see also 2.1.4) of the system, like motivating, helpful, fun, aesthetically pleasing, etc. The goal of an action should explain how the system intends to provide assistance its user, either by providing helpful information, or by alerting and reminding.

[49] http://www.upnp.org/ visited 22.07.2009

6.3.3.4 *Location*

Activity Theory also helps to analyze and model the typical situational context of activities that is important for the design of context-aware systems (4.1.5.1). Intelligent, proactive environments consider the current situation of their user and try to recognize activities from sensor readings before they try to give recommendations (4.2.3). Therefore the system requires contextual knowledge about activities and actions.

The notion of location is the key to any location-awareness and its importance has already been discussed earlier in this thesis (see section 6.1.2). We also argue that the possible locations, where an activity can be performed, depend on the location of the required artifacts. If the artifacts are mobile, it makes sense to attribute a list of possible locations to an activity, where it can or should be performed using these mobile artifacts. The activity model shall refer to any subset of the symbolic location identifiers that have been defined in the previous step.

6.3.3.5 *Time*

Allen and Ferguson (1994) have introduced a temporal logic that formally represents time as intervals I_i and 13 qualitative relations among them, such as (I_1 *after* I_2) and (I_1 *during* I_2). However, modeling the temporal dimension of an activity is not quite as trivial as one might think. Whereas numbers and intervals are good for precisely measuring time in milliseconds or whatever unit, they can't express approximations like "we meet around five" or "we will come for a visit in the afternoon". In order to make fuzzy statements, we need an ontology that defines temporal terms that are more appropriate, like "in the morning". Furthermore, the duration of activities, actions, and operations matters for activity recognition. Here the same problem appears: Typically, opening a bottle takes only a few seconds, but sometimes it can take much longer if many trials are needed. It does not seem useful to define a specific duration, and one could also argue about a minimum or maximum, as it depends on the individual user. A comprehensive discussion of spatiotemporal reasoning for smart homes is given by (Gottfried et al., 2006).

6.3.3.6 *Formal Representation of the Activity Model*

We will now give a formal representation that defines the proposed structure of an activity model, according to the categories defined in the sections above. We ground our definition on a set of symbols Ω that represents the ontology. We assume that the ontology consists of four branches that comprise physical objects (including persons and all artifacts), actional elements, spatial elements (like buildings and rooms), and temporal elements.

$\Omega = \Omega_{\text{physical}} \cup \Omega_{\text{actional}} \cup \Omega_{\text{spatial}} \cup \Omega_{\text{temporal}}$

We can now conceive the categories for subjects, artifacts, objectives, location, time, and instrumentation as subsets of the four branches of the ontology.

Subjects $\mathcal{S} \subseteq \Omega_{\text{physical}}$

Roles $\mathcal{R} \subseteq \Omega_{\text{actional}}$

Physical Artifacts $\mathcal{P} \subseteq \Omega_{\text{physical}}$

Needs $\mathcal{N} \subseteq \Omega_{\text{actional}}$

Goals $\mathcal{G} \subseteq \Omega_{\text{actional}}$

Preconditions $\mathcal{C} \subseteq \Omega_{\text{actional}}$

Locations $\mathcal{L} \subseteq \Omega_{\text{spatial}}$

Time $\mathcal{T} \subseteq \Omega_{\text{temporal}}$

Instrumentation for human-environment interaction (sensors and actuators) $\mathcal{I} \subseteq \Omega_{\text{physical}}$

We can now define sets of activities $\mathcal{A}$, actions $\mathcal{B}$, and operations $\mathcal{O}$ as 6-tuples over the sets $\mathcal{S}, \mathcal{R}, \mathcal{N}, \mathcal{G}, \mathcal{C}, \mathcal{P}, \mathcal{L}, \mathcal{T}$, and $\mathcal{I}$. For the definition of the tuples, we require the following elements and sets:

Subject $s \in (\mathcal{S} \cup \mathcal{R})$, precondition $c \in \mathcal{C}$, goal $g \in \mathcal{G}$, need $n \in \mathcal{N}$, set of artifacts $P \in 2^{\mathcal{P}}$, set of locations $L \in 2^{\mathcal{L}}$, set of times $T \in 2^{\mathcal{T}}$, and set of HEI-instrumentation $I \in 2^{\mathcal{I}}$

Now we can define:

Set of operations $\mathcal{O}$ as tuples $o=(s, c, P, L, T, I) \in (\mathcal{S} \cup \mathcal{R}) \times \mathcal{C} \times 2^{\mathcal{P}} \times 2^{\mathcal{L}} \times 2^{\mathcal{T}} \times 2^{\mathcal{I}}$

Set of actions $\mathcal{B}$ as tuples $b=(s, g, P, L, T, I) \in (\mathcal{S} \cup \mathcal{R}) \times \mathcal{G} \times 2^{\mathcal{P}} \times 2^{\mathcal{L}} \times 2^{\mathcal{T}} \times 2^{\mathcal{I}}$

In order to represent the hierarchical structure between actions and operations, we need the following relation:

Relation *Operation-Of* $R_B \subseteq \mathcal{O} \times \mathcal{B}$

Finally, we can define activities and their composition of actions:

Set of activities $\mathcal{A}$ as tuples $a = (s, n, P, L, T, I) \in (\mathcal{S} \cup \mathcal{R}) \times \mathcal{N} \times 2^{\mathcal{P}} \times 2^{\mathcal{L}} \times 2^{\mathcal{T}} \times 2^{\mathcal{I}}$

Relation *Action-Of* $R_{\mathcal{A}} \subseteq \mathcal{B} \times \mathcal{A}$

For completeness, we can define a relation $R_{\mathcal{V}}$ that denotes a set of virtual counterparts v of an artifact $p \in \mathcal{P}$: $R_{\mathcal{V}}(p, v) \subseteq \mathcal{P} \times \text{URI}$

As a result, we define:

Activity Model $\mathcal{M} = (\Omega, \mathcal{A}, \mathcal{B}, \mathcal{O}, R_{\mathcal{A}}, R_{\mathcal{B}}, R_{\mathcal{V}})$.

If the model should be able to represent sequences of actions and operations instead of unordered sets, additional successor relations $R_{\mathcal{SB}} \subseteq \mathcal{B} \times \mathcal{B}$ and $R_{\mathcal{SO}} \subseteq \mathcal{O} \times \mathcal{O}$ could be defined.

The similar structure of the sets $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{O}$ as 6-tuples suggests representing the model $\mathcal{M}$ in a tabular layout, as shown in Table 5. The table also shows an example action that is taken from the introduced scenario above, where the man shaves his beard in the bathroom in the morning around 6am-7am using an electronic shaver. If we assume that the shaver has a digital product memory, it would include information about the battery and blade status. An intelligent environment could read this information to occasionally remind the user to recharge the shaver and to buy new blades. In an AAL scenario, turning the electronic shaver on would be a clear indication that the user is in the bathroom and preparing for the day.

Table 5: The Activity Model shown as a tree-table structure.

Activity/Action/Operation	**Subject**	**Motive/ Goal/ Prec.**	**Physical Artifacts**	**Virtual Counterp.**	**Locations**	**Times**	**HEI**
Activity a_1	..	..	..	..	..	..	..
Action b_{11} “Shaving”	e.g. Man	Body care	Electronic Shaver	Battery status	Bathroom sink	around 6-7 am	On/Off status
Operation o_{111}	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..
Operation o_{11o}	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..
Action b_{1m}	..	..	..	..	..	..	..
Operation o_{1m1}	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..
Operation o_{1mp}	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..
Activity a_n	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..

6.3.4 Interaction Design

The goal of the fourth step is to enhance the activity model to an interaction model that describes how the instrumented environment provides assistance to its users on a conceptual level. To get started, the activity model shall be analyzed step by step in order to identify actions and operations that are likely to require assistance. Next, a choice of appropriate conceptual interaction models and paradigms needs to be made, considering usability and user experience goals. Then one must decide for appropriate sensor and actuator technology and define the actions and operations that shall be executed by the system.

6.3.4.1 *Analyzing Activities for Potential Assistance*

One reason for spending the effort to create the activity model of the current practices is to make the designer aware of details on the operational level that might easily be overlooked otherwise. Using the model, the designer can effortlessly analyze each action and consider cognitive and physical aspects of the user that might lead to a failure. Regarding cognitive aspects, necessary actions might be forgotten, or the required knowledge and information might not be available to the user. In case an action already involves interaction with technical devices, the subject might need instructions on how to operate the device. Manual operations can also be difficult to perform for elderly subjects that are suffering from physical impairments, e.g. tremor or weakness. Likewise, reading small print or computer displays without glasses is a typical handicap that should be considered.

In case that the new system does not focus on an incremental improvement of current practices, but rather aims towards a disruptive innovation, an activity model can still be helpful to describe new activities that are introduced with the system. The model can be applied for a structured representation of informal scenarios, which have been introduced by Carroll (2000) in the context of scenario-based design, see also Section 2.1.4. The detailed model can help to discover possible breakdowns that are introduced with technology, considering the experience of the target user group.

Once candidates for a possible failure of an action or operation have been identified in the activity model, it is the designer's job to identify the reasons for that by considering the cognitive and physical aspects of the activity, and to choose assistance options, such as reminders or automation, that will mitigate the risks. In order to prevent the user from failure at the identified weak points, one must figure out how the assistance system could recognize the situation and intervene and/or support the user. We will introduce design principles and guidelines to systematically find answers to this question at the end of this chapter.

We have reviewed in Section 2.1.1 the following conceptual interaction models to choose from: instructing, conversing, exploring and browsing, manipulating and navigating, and implicit interaction. For Instrumented Environments, implicit interaction is the most interesting. However, it requires sensors to recognize the situational context. We will now discuss how the combination of activity model and environment model can help to decide which sensors can be applied and where they have to be placed for best coverage.

6.3.4.2 *Choosing Sensor Input and Actuator Output*

A wide range of input and output devices is available for human-machine interaction today, which spans a large design space for intelligent environments. The choice for modalities determines how the interaction with the system will look and feel and is crucial for the usability and acceptance of the system. Hence it is important for the designer to know the pros and cons of different technologies in order to make a reasonable decision that fits to the requirements. The most frequently used input methods today are based on touch, like keyboards and touch-enabled screens, which are particularly useful for a manipulating and navigating style of interaction. Mobile devices are increasingly often enhanced with motion and orientation sensors, for example to automatically align the screen horizontally or vertically. Speech recognition however is ideal for instructing and conversing. Computer vision is a non-intrusive approach to sense what is going on based on video streams. However, whereas cameras and microphones are almost ubiquitous and cheap, speech and image recognition algorithms are still prone to noisy input data and difficult to use outside controlled lab environments. Finally, systems are often supplemented with simple environmental sensors that measure temperature, sound pressure, or light level. Location sensing requires a system made of sender and receiver; usually, technologies like ultrasonic sound, active or passive RFID, infrared, and ultra-wideband radio are used to measure signal strengths or runtimes.

An interesting aspect of the activity model analysis is to identify existing interaction between the user and the environment that can be evaluated as input for the assistance system. A good example is to measure the electric current of appliances to infer their mode of operation. Non-electric artifacts might be equipped with sensors that recognize changes in motion. For the future, network enabled appliances can be expected, e.g. based on standards like Wi-Fi 802.11, ZigBee, Powerline, URC (UPnP), and the proprietary @home technology. As a side effect, sensing manual interaction with an artifact always allows the system to infer the presence of a user. In the special case of a single user scenario, the recognition is possible with high confidence.

On the other hand, for the system to take any effect, information needs to be conveyed to the user, and the same question for modalities arises. Currently the most prominent output medium is vision, either by TFT displays, projection, or print. Displays can be distinguished between personal devices, public displays (also called digital signage), and ambient displays. Regarding audio, speech synthesis complements speech recognition for conversing. Whereas touch is most important for input, it plays only an inferior role for output. For example, visually impaired users read from Braille displays. Motion is more common for output, in particular to notify the user through vibration, and force feedback is well known for interaction with virtual realities. In instrumented environments, motors play an important role to manipulate the physical configuration, for example by opening and closing doors and windows, and adapting the furniture like raising or lowering a table to fit the user. Complementary to environmental sensors, heating and cooling systems can control the temperature.

The problem of choosing proper hardware for interaction is evident from the examples above. At the moment, no comprehensive source of information about interaction technologies is available. Some devices, such as touch sensor panels, can only be found at distributors for the embedded computing market, while others, like web cameras, are targeted for the consumer market. It would be a great advantage for interaction design to have access to an ontology that provides a browsable collection of technologies. Therefore the ontology that is used for the design process shall include a taxonomy of sensors and actuator concepts and instances of hardware that is available on the market.

6.3.4.3 *Placing Devices into the Environment*

Once the type of hardware has been chosen, the problem of physically placing it into the environment arises. Sensors and actuators only have a limited range of sensitivity or effect, in particularly optical devices have a limited field of view that needs to be considered, yet it is difficult to estimate without proper tools. The advantage of having a virtual environment model is that the instrumentation can be precisely planned and the number of required sensors and actuators can be figured out.

Finally, the chosen human-environment interaction technology shall be attributed to the according actions or operations in the activity model.

6.3.4.4 *Modeling System Interaction as Activity*

We have already seen in Section 6.3.3 how to model the ongoing activities of a domain without user assistance. For the modeling of assistance, the choice of sensors and actuators for the interaction between the user and the environment can be additionally represented in the activity model $\mathcal{M}$ as instrumentation I.

According to the stages of user activity (see 4.2.1) interaction takes place in turns between the user and the system. Whereas in the function-oriented computing model the user always initiates an (inter)action, proactive and mixed-initiative systems can also start the interaction with the user.

Activity Theory originates from psychology, so it assumes that subjects are human beings. However, in the domain of HCI it makes sense to extend this viewpoint and allow subjects to be any rational agent, including user assistance systems. Under this assumption we can model the dialogue between the user and the system turn-by-turn. The sequence of actions can begin with user-initiated as well as system-initiated turns.

In case of explicit user-environment interaction, the instrumentation part of the activity or operation refers to some input device, such as a keyboard or touchscreen.

In case of a system-initiated action, the instrumentation refers to both: i) the sensors that provide the necessary input about situational context; and ii) the output channels used to convey information to the user, and the actuators that control and manipulate the environment. Typical goals of an assistance system would be to:

- Provide required information to the user (either proactively or on request);
- Suggest how to proceed with an ongoing activity to achieve a (higher-valued) goal;
- Externalize information or processes (i.e. by the recording of information);
- Alert the user on events;
- Remind the user of suspended activities;
- Suggest the user better ways to achieve the same goal;
- Offer the user to finish a task on his/her behalf.

The information that is presented by the system will be available in digital form, so it can be represented in the model as a virtual artifact by its URL.

6.3.5 Implementation

The contextual knowledge about activities and actions can be applied for the implementation of activity recognition components that consider the current situation of their user in order to proactively assist them. The geometric model can be reused for indoor positioning, if navigational beacons are modeled. The spatial references from the activity model can also be utilized for location-awareness. The concepts from the ontology can be applied to uniquely denote parts of the domain and to share knowledge between applications. Furthermore, multilingual ontologies can support the localization of software and its documentation.

6.3.6 Simulation and Evaluation

In this section we will discuss the applicability of a virtual environment model for the simulation and evaluation of a system design. Instrumenting a real environment with hardware is work and cost intensive, and often implies irreversible damage like drilling holes in walls to mount displays. Sometimes testing on site is not possible at all because of construction work in progress. The simulation of the environment in Virtual Reality (VR) aids the designer to verify their concept immediately and efficiently. Important aspects are the visibility of displays and the range of sensors and actuators. We have also already mentioned the importance of evaluating early prototypes in a user-centered, iterative development process (see section 6.1.4). Virtual environments can be used to let the target users experience the system at an early stage in order to verify that the design concept meets their expectations and requirements. Finally VR helps to communicate the benefits and required hardware effort to the client and to eventually foster a discussion about alternative solutions in order to find a tradeoff between cost and performance. From the perspective of research projects, VR is a great opportunity to visualize new ideas for proposals and publications, and to share the experience of an environment with other researchers.

6.3.6.1 *Simulating Digital Signage*

Public situated displays or digital signage networks have great potential for user assistance in intelligent environments, and several applications have been published in O'Hara et al. (2003). Public displays can be used to complement personal devices such as mobile phones and PDAs. These small devices are always available to the user, but their display resources are strictly limited due to the constraints of their form factor. Whereas public displays offer large screen spaces, their visibility is restricted to certain locations. However, this restriction can be beneficial for presentations that are designed for a specific spatial context.

Traditional signage applies their limited visibility to convey context-related information, such as sales offers, opening times, room occupancy, but also local rules and prohibitions. A sign that reads "*no smoking*" for example is not intended to have a global effect, but only to stop people from smoking inside the area where they can actually read it. In addition, signs have a fixed orientation with respect to their surrounding environment, so that they are especially useful to assist us in our wayfinding tasks.

Electronic displays, even when they are non-interactive, offer more flexibility and can adapt their content to the situation. In intelligent environments they can even sense the users and adapt their presentations according to the users' profiles and cognitive resources. For example, an arrow might dynamically change its direction according to the goals of each individual user.

However, the design of signs and displays poses difficulties since the designer has to consider the visibility of the displays from various viewpoints, and the interpretation of directions heavily depends on their viewing angle. We all might have experienced traffic situations where it is unclear whether the sign tells us to turn left or to go ahead. Personalized information is yet more difficult to design, since the presentations are automatically generated and have to be evaluated for a variety of events and situations, such as different user profiles and goals.

Virtual Reality (VR) allows the designer to quickly place and view displays from various perspectives, and to evaluate their visibility and content in relation to their surrounding building structure and furnishing. Furthermore, the position of the avatar is always known in VR, so that it takes little effort to simulate adaptive signage that responds to the presence and identity of simulated users.

6.3.6.2 *Simulating Physical Aspects of Instrumented Environments*

Besides the visibility of displays, also the placement of projectors and cameras needs to be evaluated considering their field of view. For example, a typical question would be how many cameras are required to completely cover an area of interest.

If it comes to home automation or robotics, we have to consider the physical aspects of artifacts like automatic doors, shades, and height-adjustable tables. It is notable that the concept of user-adaptive systems does not only apply to software products, but it can also be transferred to smart furniture. Good examples are electronic seats in a car that automatically reconfigure to fit the driver, depending on the key used. In the context of Ambient Assisted Living the goal is to physically adapt the environment and furniture to the needs of the user. If someone shares a flat with a disabled person sitting in a wheelchair, the furniture needs to be height adjustable.

Figure 64 shows an advanced simulation setting, where the geometric environment model is instrumented with virtual counterparts of sensors and actuators that are controlled by an actual implementation of the assistance system. The communication between the virtual instrumentation and the application logic can be realized by a blackboard architecture or virtual bus. The virtual sensors can be implemented to deliver predefined values for testing, or respond to (explicit or implicit) actions of an avatar that is controlled by the developer in VR. For example, virtual proximity sensors could respond to the avatar position, and virtual buttons could afford to be pushed by the developer's avatar.

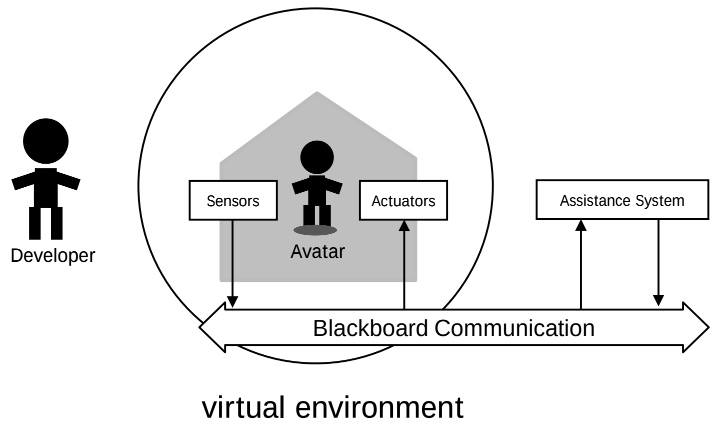


Figure 64: Simulating a design concept in virtual reality.

6.3.6.3 *Evaluating the Design by Target Users*

The idea of user-centred design is to incorporate the target users in the design process from the beginning of a project. So once a design concept has been made, it should be presented to the users to receive some feedback. We have already mentioned the evaluation techniques of demonstration facilities, living-labs, and In-home experience for user feedback (see 6.2.1). All are quite expensive regarding time and costs, so for the initial presentation of the design during the first iteration of the development cycle, virtual environments seem to be a promising alternative. We suggest that the same virtual world that is used for design and simulation should be reused for evaluation. In that case, only a software prototype needs to be implemented which is connected to the virtual environment. Location-awareness can be easily simulated by accessing the avatar position coordinates in VR without the need to set up a real indoor positioning system.

This strategy will help to reduce the effort for instrumenting the real environment during the initial phase of the development process. It is also the first step towards a dual reality concept that we will explain in the following section.

6.3.7 Monitoring and Manipulating the Real Environment in 2R

Figure 65 shows the concept of Dual Reality (2R) for instrumented environments based on blackboard communication architecture, like *SUPIE*. On the left hand side, the real world environment is depicted, including the sensors, actuators, and the user. In the example, three sensors send their readings a, b, and c to the blackboard. Two actuators read their control parameters *p* and *q* from the same blackboard. The application logic of the assistance system can be thought of two functions *p(a, b, c)* and *q(a, b, c)* that control the actuators depending on the sensor readings. According to the *GOAL* design method we assume to already have a virtual environment model that represents the building structure and virtual counterparts of the real sensors and actuators.

We have seen in the previous section how the virtual system can be used for simulation and evaluation, before the actual environment is realized. Once the real environment has been instrumented accordingly, both realities can be connected to form a so-called Dual Reality, as introduced in Section 1.4, Figure 2. The virtual model can be used to visualize the state of the real hardware sensors and actuators. In that case, the virtual counterparts of the sensors visualize the actual readings from the real environment for inspection. Yet the active implementation of the virtual sensors from the simulator can still be used to manipulate the real environment, for example to open a real door by pushing a virtual button with the avatar. However, conflicts between the state of real sensors and their virtual counterparts need to be resolved or avoided by design. For example, stateful rocker switches need to be replaced by push-buttons that only send events when they are pushed.

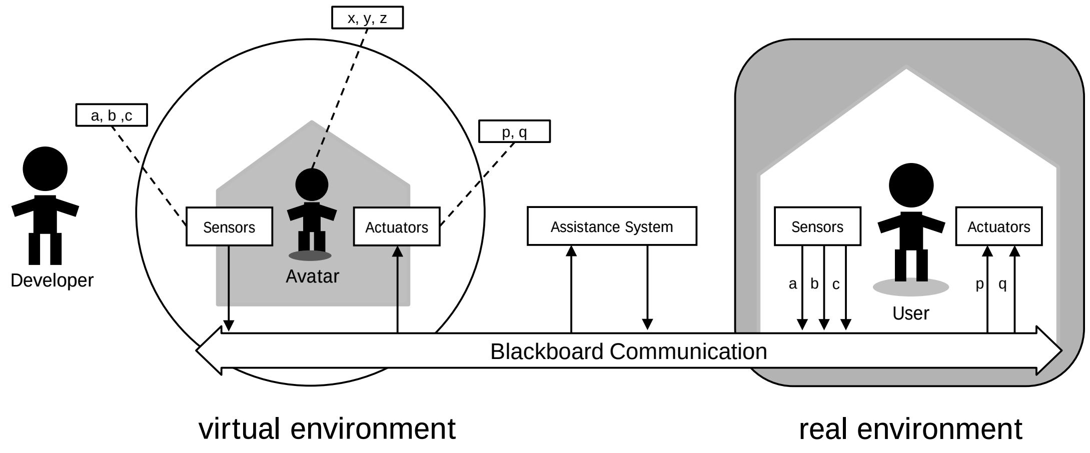


Figure 65: Dual reality allows the developer to monitor and manipulate the real world in VR.

In the following we will discuss how the dual reality concept can be applied in four scenarios: care taking, logistics, and debugging through inspection and user feedback.

6.3.7.1 *Care Taking*

Whilst the primary goal of Ambient Assisted Living (AAL) technology is to support independent living at home, it also aims at reducing the care taker's workload by providing them with detailed, up-to date information about their patients. Today, in case of a serious fall, it can take up to 24 hours until the care taker will find their patient lying on the floor, so the patient has to suffer from unnecessary pain and side-effects like dehydration. Whereas video observation is a quite effective means to remotely monitor the activity of elderly users, it is still time consuming for the care takers to watch the video streams in order to recognize emergency situations. Furthermore, 24h video observation is ethically difficult as it massively violates the users' privacy. Instrumented environments employ sensors to provide qualitative information about the users' situation, which could be visualized in VR using a dual-reality model on a more abstract level than video recordings.

6.3.7.2 *Remote Surveillance and Control*

We have already introduced the SmartFactory environment (see Section 3.1.4) and a scenario where Zühlke (2008) describes how mobile devices can support workers at an industrial production plant in their repair jobs. Based on an indoor positioning system, the mobile terminal adapts its user interface to the machines nearby the user.

In a 2R environment a virtual model can be used to visualize the position of the workers in real time which allows the supervisor to assign the nearest worker for a pending maintenance job in order to reduce delays that are caused by walking distances. Furthermore, the service staff could remotely inspect and control the machines in a virtual model. By virtually walking through the factory model, they can apply their spatial knowledge from the real world to navigate to the user interface of a certain machine in 3-D. In certain situations, this might be more cognitively adequate than symbolic navigation through lists and menus.

The same principles could be applied to retail scenarios, where a branch manager could monitor the shelves to see if enough items are in stock, and customers could find products in a 3-D online shop by applying their spatial knowledge of the real store.

In domestic environments, virtual models could allow users the remote surveillance of their homes; they can check for open windows and remotely control the heating in order to warm up the house before their arrival when returning from a vacation. In an AAL scenario, users could monitor and control the homes of their parents.

6.3.7.3 *Visually Inspecting and Debugging Instrumented Environments*

It must be anticipated that intelligent environments will suffer from similar reliability issues as desktop software applications, since they rely on complex software components for the realization of intelligent behavior that communicate with hardware components from different vendors. Hence programming bugs are likely to occur that lead to disturbance, ranging from minor usability bugs to a complete failure of the environment. However, the users of automated environments can’t simply ignore errors, or, in the worst case, just abandon the system. If doors do not open and lights do not turn on as expected, the user will literally be in the dark. Formal methods can be applied to develop and verify software in order to avoid faulty code, but they are costly and require expert knowledge. Furthermore, they can only prove that a piece of software complies with its specification; wrong assumptions in the system requirements and design can’t be resolved by formal methods.

Besides software, instrumented environments also depend on hardware so that the relationship between cause and effect of a failure is a very complex one. Let’s consider a rather simple scenario, where a wireless light switch is used to control a lamp. Now assume a typical failure: the user is unable to turn on the lamp by pushing the button. Due to the complex architecture of IEs, the problem could occur on four levels:

- **Application**: The control program might have exited with an exception that was caused by a software bug at runtime;
- **Network**: The network might be misconfigured or disrupted by radio interference;
- **Controller**: Either the switch or lamp microcontroller might be damaged or suffer from power failure;
- **Physical**: The tungsten filament or the contacts of the switch might be broken.

Hence it seems absolutely necessary to develop debug tools that provide insight into the actual state of sensors and actuators similar to software development environments. Typically, raw console output of data packets is not human readable: too many packets are sent per second, and the data is bitwise encoded and may consist of multi-digit identifiers. Hence graphical tools are important to inspect the actual status of sensors and actuators, similar to a voltmeter on the physical layer. Does the switch really sends an 'on' message, and does the lamp really receive an 'on' signal? In case that the switch sends no signal, it is likely to cause the error (on physical or controller layer). In case that the lamp receives a signal, the light bulb is likely to be broken; otherwise, the problem might be caused on the application layer. The dual-reality VR model can provide an ideal environment to trace system errors.

6.3.7.4 *Incorporating Quality Feedback from the Users*

It is a typical feature for today's software products to include quality feedback assistants that automatically report failures to the vendor. Microsoft's error reporting technology[50] is a good example. If a program closes unexpectedly because of an error, Windows XP asks the user to send an error report (see Figure 66). Microsoft gathers the error reports that customers send, analyzes them, and uses the aggregate information to determine which problems are causing the most suffering. The error-reporting information includes information about the system configuration and traces of the system memory and enables Microsoft developers to identify the most serious bugs and resolve them first.

This user-generated feedback loop extends the development cycle even beyond the deployment of a software product in order to trace errors in the implementation and thus helps the vendors to improve the reliability of their products.

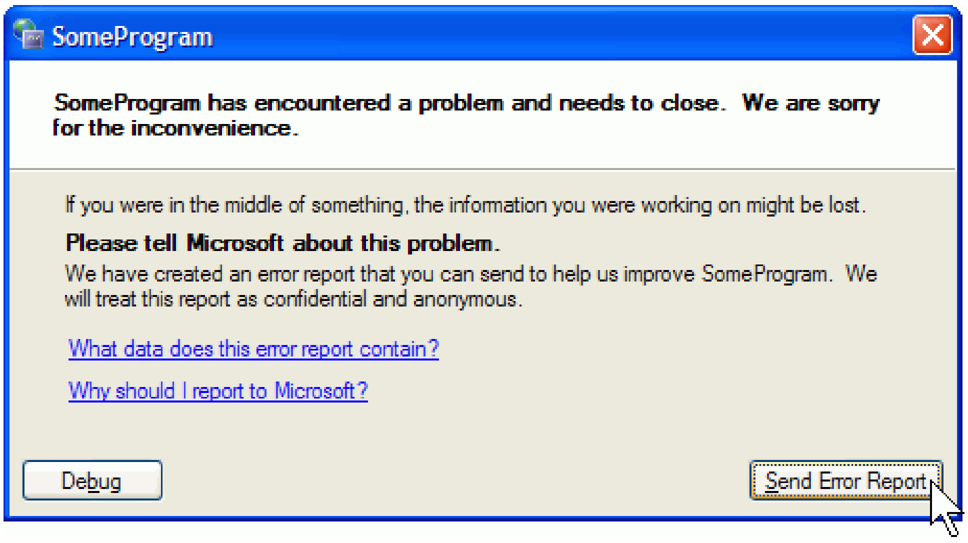
We argue that a design method for intelligent environments should also incorporate quality feedback reports from the users after the system has been deployed.

It is often difficult or even impossible for the user to describe the exact conditions that cause faulty system behavior, since the system "output" depends on the input of many sensors and the overall system state. Plane crashes are a good example – the pilot can often not tell the engineers which circumstances caused the accident. The flight data recorder device has been invented to overcome this problem by continuously gathering and recording information about the state of the plane. If a crash happens, the cause can be researched afterwards from the recordings, similar to the analysis of software error reports.

In the BMBF-funded DFKI project SPECTER[51] a context log (Kröner et al., 2006) has been developed that logs contextual information about the user's actions as perceived by various sensors in instrumented environments. This log works as a long-term memory of the environment and also includes system actions. Personal journals grant the users access to perceptions that regard their interaction with the environment and allows them to learn about the course of events through introspection. These context logs allow to serve as error reports in IEs.

[50] http://www.microsoft.com/windowsxp/using/setup/support/error.mspx

[51] http://www.dfki.de/specter/


SomeProgram

SomeProgram has encountered a problem and needs to close. We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.

We have created an error report that you can send to help us improve SomeProgram. We will treat this report as confidential and anonymous.

What data does this error report contain?

Why should I report to Microsoft?

Figure 66: The well-known system prompt asking the user to send an error report to Microsoft

Having a virtual simulator of the environment at hand would allow developers to reproduce faulty system behavior that has been reported by the users. Similar to existing debugging tools for software applications, such a simulator should afford the inspection of the internal system state and sensor readings, based on the recordings from the error report. In our proposed dual reality architecture, the concept of a black box can be realized by recording data from the real environment until the user experiences a situation of faulty or undesired system behavior (see Figure 67). In that case the user can report the failure to the developer. The actual recording may be sent to the developer, who can playback the recording in the virtual environment in order to reproduce the situation and try to and analyze the cause for the failure. This strategy is important to be able to reproduce errors without affecting the normal operation of the environment and helps to save travel expenses.

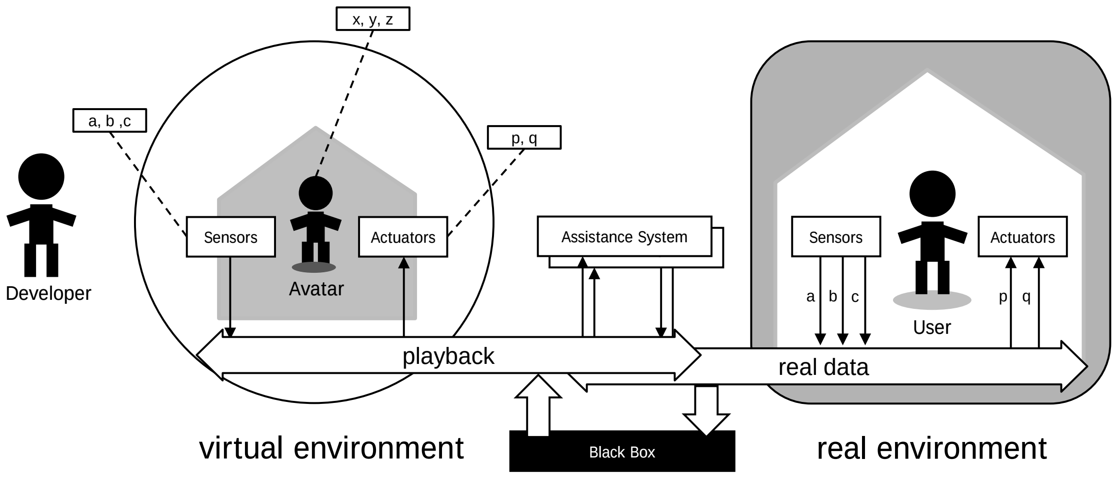


Figure 67: Recording faulty system behavior with a "black box".

6.4 Modeling and Design Considerations

In this chapter, we introduced the GOAL method and described how the geometric environment model and the activity model can support the designer of an assistance system. In order to keep the models concise and focused on relevant aspects, the designer first has to make some important decisions about the scope of the system that is to be developed. In the following, we will discuss how to find the right level of abstraction. Then we will discuss different kinds of navigational aid and the implications on the spatial model's extent and level of detail. Considering interaction design, the most important decisions are related to conceptual models, interaction paradigms, and the choice of modalities and devices. Finally, we will discuss how to deal with conditions and repetitions in the activity model, and investigate the relationship between the system and its users.

6.4.1 Finding the Right Level of Abstraction

Most activities turn out to be quite complex if we start to analyze and model them bottom-up from an operational level. On the other hand, a top-down approach is not much easier if we really try to consider the whole chain of motives and needs behind simple tasks. According to Activity Theory and the principle of division of labor, the writing of a word document is motivated on a higher level to earn money and buy some food. Whereas such top-level knowledge might be helpful for the design of some activity-adaptive systems, as described in Section 4.3, probably not all applications will take a serious advantage out of it. The scope of the activity model should fit the system's purpose and applicability of context. So in order to begin with activity modeling, the first question should be which operations need to be considered and which activities and motives should be modeled at the top-level. According to the principles of Activity Theory, activities can be distinguished by their objectives, and the actions and operations can be distinguished by their cognitive effort (see 2.2.2).

Similarly to Object-Oriented Analysis, parts of the real world can be modeled in the ontology either as classes or instance. Hence a second question is whether it is necessary to be able to instantiate multiple objects of the same class/concept. For example, in a shopping assistance scenario (see 5.2.2) it might be sufficient to represent product types, like a certain camera model, as instances. If the system should be able to keep track of individual cameras, e.g. by their serial numbers or product memory, the product type must be represented as class, so that each individual camera can be instantiated from that class with its own properties.

The same consideration applies to the subject and artifact of an action in the activity model. Subjects can either be persons, like *Peter*, or roles, like *user*. Likewise, artifacts can be described by individual instances, like *Peter's dishwasher*, or generalized, e.g. *dishwasher*.

We will now suggest three design dimensions to classify assistance systems according to the type of activities that they support and discuss the above questions in this context (a summary of all design dimensions will be given at the end of this Section in Table 6):

- **Strategic, tactic, or operational level**. Assistance on a strategic level concerns long-term goals and deals with the selection and scheduling of activities. For example, a to-do list helps the user to keep track of their tasks and a calendar helps to remember events. Typical multi-tasking related issues are notifications and reminders that trigger the beginning of tasks, and coordinate suspending and resuming activities. Assistance on a tactical level usually suggests the best actions towards the current goal, or automatically performs actions on behalf of the user. An intelligent system should also react to contextual changes. Assistance on the operational level mostly offers convenience or externalization functions, e.g. quick-dialing, help through tool tips, and means to reduce errors, e.g. a spell checker. On a physical level, artifacts can ease tedious manual operations, e.g. an electric tooth brush.

- **Dedicated purpose** or **general purpose**. The scope of a user assistance system can range from a very specific domain to a general-purpose tool.
- **Help system** or **assistance system**. Help systems explain the user how to use a system, e.g. how to use its user interface and the available functionality. Assistance systems explain the user how to solve tasks in the real world, e.g. by providing route directions. Assistance systems may include help components, for example a navigation system can embed online help that explains how to pan and zoom the map.
- **Ambient Assisted Living.** Nehmer et al. (2006) have classified assistance systems in the AAL domain as follows: i) Emergency treatment; ii) Autonomy Enhancement; and iii) Comfort services.

For systems that are focused on the strategic level, e.g. Ajikata and Kobayashi's (2009) Task Manager, activity modeling should start top-down with the user's motives rather than goals. If the system is intended to support the tactic level, tasks and goals might be more relevant than top-level activities. In that case, it might be better to start the modeling bottom-up, and tasks may be represented as activities instead of actions.

Systems with a dedicated purpose, e.g. custom-made control software for an individual smart home, or help systems for a specific application, are most likely modeled best with individual instances and as concrete, user-specific activities. For general purpose (off-the shelf) systems it is important to generalize specific observations that were made by informal elicitation techniques, as described earlier in Section 6.2, into behavioral patterns.

6.4.2 Modeling for Navigational Assistance

Regarding the spatial environment model it is also important to choose a reasonable level of abstraction from the beginning, so that it represents all necessary information with minimized modeling effort. This decision depends on the spatial extent and the level of detail. The relevant physical extent can range from continents (car-navigation) to rooms (e.g. locate books in a library), and the granularity of the modeled entities can range from cities and streets (outdoor scenarios) to rooms and shelves (instrumented environments). Furthermore, it is important to consider if spatial knowledge about services in the environment is required, and which services are really relevant for the user.

These factors mostly depend on two design dimensions of navigational aid:

- **Micro-navigation** or **macro-navigation**. Different models, positioning technologies, and modalities are required to provide adequate navigational assistance on a local scale (vista space) and large scale (environmental space), see 2.3.3 for more details.
- **Navigation** or **exploration**. Spatial tasks are mostly referred to by the term *navigation*. According to Wikipedia, navigation actually is a *process of planning, recording, and controlling the movement of a craft or vehicle from one place to another*. This process implies that the user has a clear goal and destination. In

contrast, tourists often explore their environment without such a goal, but with the motivation to experience the *sense of place*.

Navigation systems typically address the case of macro-navigation, where the destination is outside the perception of the user, and route knowledge is required at decision points. In outdoor scenarios, this requires a large-scale map of streets, and satellite positioning system. Indoor navigation usually takes place in a smaller scale, like a building, and requires alternative positioning technologies, like WiFi-based systems. Micro-navigation addresses the problem to localize a certain item in the user's vista space. To be effective, this requires a precise, small-scale model of the room, and information about the location of items in 3-D. This can be achieved with positioning systems like the ultra-wideband based Ubisense, or the ultrasonic Cricket system. Alternative solutions are optical tracking systems, like the AR toolkit for example.

Whereas navigational tasks require concise and clear instructions, explorative tasks are better supported by an unobtrusive and minimal attention user interface. The latter tasks also benefit from survey knowledge, like you-are-here maps, and location-based services. A good example is the context-aware tourist guide presented by Cheverst et al. (2000).

6.4.3 Interaction Design Considerations

Once the necessary activities have been identified and modeled on an appropriate level of detail, the design process proceeds with the modeling of the interaction between the user and the system. The interaction design should be consistent and easy to understand. Therefore it is important that the designer decides for a conceptual model and interaction paradigms that suit the users' needs. We have already presented and discussed interaction concepts and paradigms in Section 2.1.

Furthermore, the designer should decide for usability and user experience goals. Typical usability-related design goals are that a system is efficient to use and easy to learn, whereas experience-related goals address system properties like fun, entertaining, and aesthetically pleasing (see Figure 11 for an overview). Usability and user experience goals are often contradictory, so the designer of a system should carefully choose among them. Visually engaging presentations are necessary for a good experience, but they consume a lot of resources and thus often lack efficiency since they respond slowly to user input. The design should be appropriate for the actual situation and requirements of the user. In a museum environment, user experience goals are reasonable to catch the user's interest, whereas in an airport, some people have time pressure to catch their flight. Hence usability goals like efficiency and ease of use are more important in that scenario.

The decisions for concepts and goals are closely related to the system's overall degree of intelligence, which can be considered as another design dimension:

- **Degree of intelligence**. The intelligence of a system depends on its ability to learn and adapt itself to the user and the current situation based on commonsense know-ledge about the world. According to the classification given in Chapter 4, we disting-uish (from lowest to highest level of intelligence) between: i) function-oriented sys-tems; ii) goal-based systems; iii) user-adaptive systems; and iv) activity-adaptive systems.

Whereas function- or goal-based systems are typically designed to fit the usability and experience expectations of a specific group of users, adaptive systems have the ability to adapt their user interface design according to individual users and situations. For example, in the case of time pressure, the user interface could change from a multi-media enriched one to a simplified version that is optimized for usability.

Conceptual models and interaction paradigms also have a great influence on the perceived intelligence, usability, and experience of a system. Anthropomorphic interfaces with virtual characters are usually experienced as more enjoyable, helpful, and motivating, even though they might not be as efficient as a traditional desktop metaphor. However, users also expect a high level of intelligence from virtual characters and are easily disappointed if the implementation lacks such intelligence. In the context of ubiquitous computing and intelligent environments, particularly the concepts of implicit and proactive interaction can be applied to make a system easy to use and helpful. Tangible interfaces and organic design can also enhance the user experience and contribute to make systems fun to use, supportive of creativity, and aesthetically pleasing.

Furthermore, it is interesting to consider the level of control from the perspective of the user and the system:

- **Level of control**. Does the system only provide suggestions to the user, or can it autonomously act on behalf of the user? Vice versa: to which extent the user stays in control over an automated system?

In case that the user doesn't know how to perform some actions in a computer system, it is helpful if the system can perform them on behalf of the user. For example, computer devices often come with wizards that guide the user through a set up process and hide the original menus and dialog windows from a novice user. In the case of AAL, it might be important for physically impaired users that the environment can automate physical operations, like opening roller shutters. Vice versa, it is particularly important for automated environments that the user stays in control of the system. This requires that the user has a correct mental model of the system and that the system allows its user to manually change settings and execute functions, if the automated behavior doesn't lead to the desired results.

Once the designer has decided for appropriate concepts, paradigms, and usability goals, the interaction between the user and the system has to be modeled in detail. This implies further decisions about initiative, physical and virtual artifacts, mobility, and modalities and devices. In (Stahl, 2006a) we have proposed five questions that can guide the necessary decisions to define interaction between the user and an assistance system:

1) What are the preconditions and effects of an action/operation?

Answering this question in detail helps to identify problems and opportunities for user assistance regarding the

- Temporal aspects
- Resources (physical objects or information)
- Location of the user and physical artifacts
- State of the environment (situational context)
- State of the assistance system (system context)

Our proposed activity model distinguishes between physical artifacts and virtual artifacts (information resources) that are needed by the user to perform an action. We have already explained in Section 6.1.2 the close relationship between the location and mobility of required artifacts and the places where activities are usually performed. Therefore it is helpful to consider the level of mobility of devices and their dependency from the environment.

Dix (2000) suggests three levels of mobility within the environment: i) fixed devices that are not mobile at all; ii) mobile devices that may be moved by others (e.g. mobile phones); and iii) autonomous devices that may move under their own control (e.g. robots). Considering the Relation between devices and their environment, Dix (2000) further distinguishes between three different categories: i) free: the computational device is independent of other devices, and its functionality is essentially self-contained; ii) embedded: the device is part of a larger device; and iii) pervasive: the functionality provided by the device is essentially spread throughout the environment.

Other resources may be tickets or documents that are issued to manage processes, e.g. the passenger process at an airport is based on boarding tickets and passport checks.

2) Which actions should be mapped to interactions with the assistance system?

According to the definitions given in Section 4.2.3 one can distinguish between i) explicit interaction; ii) implicit interaction; and iii) incidental interaction. For the design of proactive systems, the activity model should first be analyzed for potential actions and operations that can be used as implicit input – often the user already interacts with artifacts; sensing these actions can substitute explicit interaction and contribute to the system's usability.

3) Who initiates the interaction, the user or the system, and what is the intention?

In the traditional model of human-computer interaction, as described by Norman's seven stages of user activities (4.2.1), the user triggers processes by the execution of functions and in turn receives a result from the system. In the context of user assistance systems, typical intentions for **user-initiated** interactions might be to: i) satisfy an information demand; ii) externalize information by making notes, e.g. adding products to a shopping list; iii) create a reminder to coordinate multiple activities, e.g. setting an alarm for a meeting; and iv) communicate or exchange documents with others. Typical intentions for **system-initiated** interactions might accordingly be to: i) convey a notification message or alarm signal to let the user begin or resume an activity; ii) recommend further actions towards task completion, and iii) perform actions on behalf of the user. Intelligent user interfaces can apply plan recognition in order to recognize the users' needs and to proactively take the initiative. In that case, the system requires additional, implicit input, which in case of intelligent environments requires sensors and middleware. Mixed-initiative systems combine user-initiated and proactive turns (4.2.3.3).

4) What is the flow of information for an interaction?

Answering this question implies to consider possible sources and destinations, which can be either virtual (databases, documents, or media streams) or physical (input devices, sensors, output devices, or actuators). The flow of information between the user and the system is a matter of suitable modalities for the given environment. Crucial factors for visual information are e.g. display size, angle, and arrangement. These factors have been investigated in a CSCW scenario by Inkpen et al. (2005). For computer vision, stable lighting conditions are highly important, since digital cameras are very sensitive to low light or sunshine. Likewise, audio (particularly speech recognition) has to be applied carefully in noisy environments. Finally, some real-life situations require hands-free operation of the system. In other situations, the user may be hindered to use a keyboard or touchscreen by dirt or gloves. Baber et al. (1999) describe a design method for wearable systems that uses a so-called modality matrix to rate all possible modality choices against the initial requirements in order to reduce the design options.

In scenarios that involve multiple users, Dix (2000) suggests to reflect the cooperative nature of advanced mobile applications by considering the extent to which the device is bound to a particular individual or group: i) personal: the device is primarily focused on supporting one person; ii) group: the device supports members of a group such as a family or business group; and iii) public: the device is available to a wide group.

Interactive kiosk devices can be considered as temporary personal devices. An ATM for example can only be used by a single person at a time. Public displays on the other hand can be seen by many users at the same time, but often lack any individual information and interaction due to the difficulties that arise from designing interfaces for multiple users. An attempt to classify interfaces for multiple users and multiple devices has been made by Kray et al. (2004), who distinguish four classes of interfaces (from single user/single device to multiple users and devices) and dimensions like group size, shareability, privacy, and distributed nature of the interface. Kray et al. (2006) have also investigated the sharing of control of situated displays between nomadic and residential users.

5) What types of actions are performed by the user and the system?

Regarding the user, one can distinguish between the following types of actions: i) physical actions, e.g. move or pickup an artifact; ii) communicative actions; and iii) cognitive actions, e.g. read, remember, recognize, recall, plan, and decide. Considering the system, typical computational actions would be to lookup information from a database, create a presentation, and planning tasks.

6.4.4 Temporal and Conditional Relations between Actions

The primary intention of the activity model is to give a survey of typical activities and their related actions and operations and to represent their spatial and temporal context. Hence the model should focus on some representative scenarios or use cases, rather than a complete enumeration of all possible combinations or conditions.

The model should not replace a library of plans that describe in detail how certain goals can be achieved. Although it can represent sequences of actions, we have intentionally left out means to represent conditions and loops. Repetitions and multiple instances of the same actions and operations should be avoided, if possible, in order to keep the model clearly represented and maintainable. The idea behind the activity model is rather to provide the necessary concepts to specify plans later during the development process, i.e. in the implementation phase. In the context of plan- or activity-recognition, actions and operations can be considered as transitions that move within a plan execution state space from an initial idle state to some goal state, as described by Schneider (2009), see also Section 4.2.3.1.

6.4.5 Relationship between the System and its Users

In this section, we will examine the relationship between the user and the assistance system in three further design dimentsions and discuss the implications for the activity model and interaction design.

- **Manual user modeling** vs. **Machine learning.** We can distinguish two different approaches towards user modeling. Some systems ask their users to manually enter their individual interests and preferences into their user model; this is also what people use social networks for. Other systems just observe what people do and apply statistical machine learning processes to build internal user models.

The manual modeling of users, their tasks, and activities can take a high effort. For example, modeling the complete range of activities that can be performed at home would be challenging project, particularly if we consider related artifacts, and details such as cooking recipes. Some argue that this approach is impractical and favor machine learning methods, particularly in computer environments, where large datasets can be collected from user input logs or other sensor data. The learned correlations between actions and artifacts can be similarly helpful to users as manually edited task knowledge. However, due to their statistical nature, some suggestions may be meaningless or even false in certain situations.

- **Active user participation.** The great success stories behind websites who let their users actively contribute to their content inspired O'Reily (2005) to his definition of the **Web 2.0**. Among others, he identifies three core principles for the Web 2.0: i) (Web) services instead of packaged software; ii) an architecture of participation; and iii) the harnessing of collective intelligence. Wikipedia articles and product reviews are good examples for user participation and collective intelligence.

Active user participation can help to overcome the limitations of statistical methods, since users contribute their knowledge and intelligence. Even in complex tasks, like the creation of the *Wikipedia* encyclopedia or *Open Street Map*, the Web 2.0 approach has proven to be successful. The idea of collective intelligence could also be applied to the modeling of everyday life activities.

- **Cooperative** vs. **Competitive environments**. The relationship between the user and the provider/vendor of a system plays an important role for the decision-making behavior of a system. Does the user of the system have similar or opposing interests as the vendor of the system?

It seems noteworthy to consider the relationship between the user and system vendor. A system will most likely be designed to represent the intentions of its provider. In case of the public (non-profit) sector, for example museums, the relationship can be considered to be a cooperative one. This means that the system acts in an unbiased and neutral manner and its suggestions represent the best choice from the user's perspective. In case of a profit-oriented environment, such as a shopping mall or amusement park, the relationship between the user and the vendor of the assistance system is of competitive nature. This situation can lead to disadvantageous results of which the user is not even aware. Sponsored search results are a well known example. Another more obscure example comes from the airport scenario. Since airports today depend on the success of their shops, certain airports intentionally retain boarding and gate information until 30 minutes before takeoff, so that passengers are likely to stay longer in the shopping area before they move on to the gate[52].

[52] Die neuen Shopping-Center (mit Flughafen-Anschluss), In: P. M. Magazin Ausgabe 5/2006, G+J Verlag.

Table 6: Summary of design dimensions for user assistance systems.

Which activities are assisted by the system	
Strategic, tactic, or **operational level**	Assistance on a strategic level concerns long- term goals, and deals with the selection and scheduling of activities. Assistance on a tactical level usually suggests the best actions towards the current goal, or automatically performs actions on behalf of the user. Assistance on the operational level mostly offers convenience or externalization functions.
Dedicated vs. **general purpose**	The scope of a user assistance system can range from a very specific domain to a general-purpose tool.
Help vs. **assistance system**	Help systems explain the user how to use a system, e.g. how to use its user interface and the available functionality. Assistance systems explain the user how to solve tasks in the real world, e.g. by providing route directions. Assistance systems may include help components.
AAL Services	Emergency treatment / Autonomy Enhancement / Comfort services.
Wayfinding assistance	
Navigation vs. **exploration**	Do the users require navigational aid, that is guidance how to go from one place to another, or do they rather require an overview or survey of places and possible activities?
Micro- vs. **macro-navigation**.	Does the user's activity require navigational assistance on a small (vista space) or large (environmental space) spatial scale?
How is the assistance provided through the system	
Degree of **intelligence**	The ability to recognize a problem and to assist the user can range from manual recovery over wizards and agents to fully autonomous services. The intelligence of a system also depends on the ability to learn and adapt itself to the user and current situation based on commonsense knowledge about the world.
Level of **control**	Does the system only provide suggestions to the user, or can it autonomously act on behalf of the user? Vice versa: to which extend stays the user in control over an automated system?
Relationship between the user and the system	
Manual user modeling vs. **Machine learning.**	Some systems ask their users to manually enter their individual interests and preferences into their user model; this is also what people use social networks for. Other systems just observe what people do and apply statistical machine learning processes to build internal user models.
User participation	To what degree are users involved in the creation of content that is presented by the system?
Cooperative vs. **Competitive** environment	Does the user of the system have similar or opposing interests as the vendor of the system?

6.5 Summary

In this chapter we have introduced a new user-centered design method called GOAL (Geometry-Ontology-Activity ModeL) that focuses on the specific requirements of developing user assistance systems for instrumented environments, particularly in AAL (Ambient Assisted Living) scenarios. We have motivated our approach to ground the method on a spatial 3-D model of the environment with the need to plan the physical arrangement of sensors and actuators and to simulate and evaluate them in virtual reality early in the design process.

By conceiving activity as triple of subject, artifact, and objective, we have investigated the relationship between activity and space and concluded that the mobility of an activity depends on the mobility of the involved artifacts. As technology progresses, more and more activities can be performed anytime and everywhere, but still cultural conventions apply, known as sense of place; they should be considered in the design of new technologies.

Since user acceptance is one of the biggest challenges in AAL projects, user-centered design processes are essential; hence we have presented a survey of elicitation techniques for capturing the users' needs. These methods usually lead to rather informal and anecdotal results, so one objective of the GOAL method is to formalize them with Activity Theory.

According to our definition, we have described the goal method in six steps. First, the physical environment is modeled in 3-D, including the building structure, the interior's furnishing, and included artifacts. In the second step, the ontology has to be extended with new concepts, including the spatial regions of the environment model. Third, the findings about the users' needs are collected and formalized in a situated semantic activity model that includes references to subjects, artifacts, places, and times, as defined in the ontology. Then we have described how this model can be applied in order to identify opportunities for user assistance in the fourth step of the method. In this context, we have pointed out how important the 3-D model is to guide the designer's decisions about the instrumentation of the environment with sensors and actuators and their placement. Finally, it is reasonable to evaluate the resulting interaction design with target users as early as possible, i.e. before the actual instrumentation is deployed. We have outlined how the 3-D model can be used for a simulation of digital signage in virtual reality, and suggested how the concept of dual reality (2R) can be employed in four different scenarios. In AAL, a 2R model could be used to monitor and control the home environment of relatives. In an industrial setting, 2R could help to localize workers and enable remote maintenance. Finally, 2R could support to debug instrumented environments and help to incorporate user feedback through error reports.

7 The YAMAMOTO Toolkit

"Everything should be made as simple as possible, but not simpler." (A. Einstein)

Modeling the physical environment is a basic requirement for the design and development of user assistance in intelligent environments, especially if navigational aid has to be conveyed to the user. Furthermore, any location-aware services may benefit from the ability to relate the location of their users and devices to features of their surrounding space. Due to the diversity of research conducted in the field of mobile and pervasive computing, current off-the-shelf products are however unlikely to meet all potential requirements for location modeling. We have faced many limitations of existing modeling standards like X3D. Most importantly, these standards are focused on visualization and do not properly support semantic knowledge that is needed for wayfinding. The modeling tools that we used in the beginning of the project *REAL* (*AC3D* by *Invis*[53]) at that time also lacked reasonable support to model buildings with multiple floors. Furthermore, interaction with the virtual worlds is limited to the built-in scripting capabilities or requires the usage of complex programming interfaces. Other shortcomings exist in the form of conversion and maintenance of geometric and geographical data through proprietary tools and data structures. Our approach is to ease the whole modeling process through the development of a new graphical editor and XML-based file format, dedicated to the specific requirements of pedestrian navigation in instrumented environments. Furthermore, we want to support the design and implementation of intelligent environments during all stages of the development process, as proposed by the *GOAL* method in Chapter 6.

Considering navigation tasks, the situation of a pedestrian trying to find a way through a city differs from driving a car, since the user is not bound to follow paths or streets. Instead users typically cross open spaces, directly following their line of sight. The model has to particularly reflect this and represent places as polygonal objects in contrast to commercial GIS (Geographic Information System)-based street map databases, which usually consist only of line segments that represent path networks as a graph data structure. These models have severe limitations for the pedestrian navigation domain on a small scale. The suggested routes will often direct the user on a detour that strictly follows the modeled paths, even if it would be possible for a pedestrian to abbreviate the route across large places. An actual example is shown in Figure 68, where the dotted lines indicate the path network and the thick line shows the abbreviated route. This can be avoided by modeling the environment as spatial regions as shown in Figure 69, where the dotted lines represent connections.

[53] http://www.inivis.com/ visited 23.07.2009

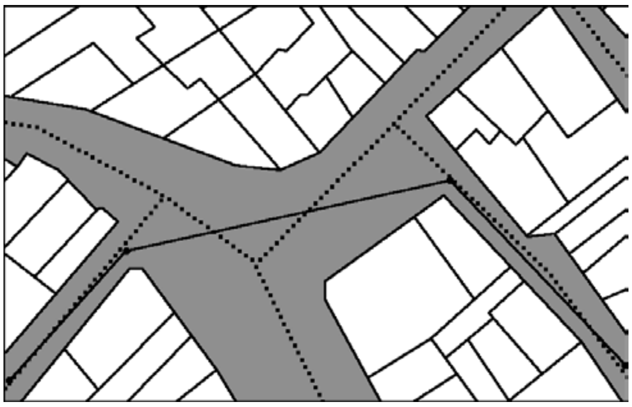


Figure 68: Pedestrian crossing a large place

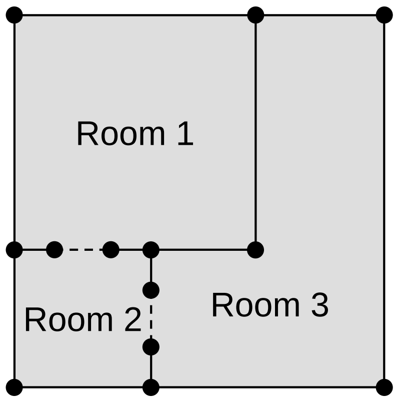


Figure 69: Modeling rooms as connected spatial regions.

As pedestrians spend most of their time inside buildings, indoor environments have to be modeled. Sophisticated architectural designs, such as the spiral ramp inside of the Guggenheim museum in New York, pose a challenge to their modeling through simplified 2-D levels, and require a solution somewhere in between the simplicity and complexity of two and three dimensional space.

We assume that we cannot base our modeling activity on a single source of information. For some outdoor areas, highly detailed aerial photographs may be available. For other areas, CAD drawings with features such as the outlines of paths and buildings may be used. Indoors however, typically there are only large-scale printouts of architectural drawings. Thus the editor should allow for a partial mapping of the environment through multiple views at arbitrary scales and granularities, and support their arrangement to form a hierarchic tree-like structure. Such a 'divide-and-conquer' strategy would also ensure scalability and maintainability.

In order to support the design of instrumented environments according to the GOAL method, it is required to be able to visualize the modeled environments in 3-D, including their furnishing and instrumentation with sensors, like RFID readers, and actuators, like displays. In this context, the modeling of navigational beacons is important for indoor positioning systems. Regarding optical devices, it is important for the design process to represent and analyze their field of view, e.g. to place a projector in the correct distance from a wall, or to choose the right lens for a camera.

Ideally, the toolkit should also facilitate the evaluation, monitoring, and remote debugging of instrumented environments in a Dual Reality setup, as described in the four application scenarios of 2R in Section 6.3.7. This implies to implement the behavior of virtual sensors and actuators and to connect them to the real environment's framework.

Finally, the toolkit should support the creation of situated semantic activity models. Therefore it is necessary that the editor supports symbolic identifiers for all geometric entities, including spatial regions, furniture elements, and interaction devices. In order to take advantage of such a hybrid model, it is important that the activity model is integrated into the toolkit's user interface. This includes the assignment of symbolic instances from the ontology to geometric entities, and the visualization of the spatial context of activities. Vice versa, the editor should support to look the symbolic counterparts of geometric objects in the ontology.

In summary, the major goals of Yamamoto are the following:

- To create a semantically annotated 2½-dimensional location model that allows for route finding in a polygonal representation (without an additional path network);
- To model the instrumentation of the environment with sensors and actuators for user assistance, i.e. the mapping of beacons used for indoor positioning;
- To allow for the virtual exploration, simulation, and evaluation of intelligent environments through an avatar interface;
- To support hybrid symbolic and geometric location modeling;
- To integrate support for the modeling of activity according to the *GOAL* method.

As for the additional subgoals, they are the following:

- To support the hierarchical modeling of areas with multiple buildings to allow for scalability and maintainability of the model;
- To store the model in a human-readable XML file format;
- To couple the virtual environment with the real environment in a Dual Reality setting;
- To keep the modeling process as simple as possible (but not simpler).

We have addressed the above goals by implementing the *Yamamoto* (**Y**et **A**nother **MA**p **MO**deling **TO**olkit) toolkit (Stahl and Haupert, 2006; Stahl, 2008), which is positioned between two-dimensional data structures, like the aforementioned GIS (Geographic Information Systems) street maps or proprietary two-dimensional location models, and professional three-dimensional CAD (Computer Aided Design) software for architects or photo-realistic visualization.

By following Einstein's motto to keep everything as simple as possible, we have intentionally reduced the expressiveness of the location model from full three-dimensional space, because CAD tools usually require a high level of experience with 3-D graphics and the modeling process involves too much effort; the designer has to manually cut out windows and doors from solid walls and has to take care about window sills, choices of door handles, or steps of a staircase. Such a high level of detail is not required for applications like route finding and navigational aid or context modeling in Ubiquitous computing scenarios. On the other hand, our model exceeds the capabilities of two-dimensional path networks so that our system is able to model streets and rooms as polygonal spaces and in multiple levels. We will introduce this 2½-dimensional model in the next section.

7.1 Geometric Location Modeling

In order to allow for a simpler and easier to learn user interface than existing 3-D modeling tools, we have reduced the degrees of freedom by half of a dimension. We will now explain what that exactly means and what are the resulting benefits and implications.

Buildings, rooms, or room-segments have to be represented as spatial regions; the former are usually physically shaped through solid walls, whereas room segments represent logical structures or areas that are determined by artifacts and activities. We use meshes of polygons as data structure for the representation, where each polygon defines the boundaries of a spatial region. Each polygon object is defined by an ordered sequence of vertices. Each vertex is located in three-dimensional Euclidean space and is represented through Cartesian coordinates as a triple of (x, y, z) values. In our data structure, vertices are indexed by integer values and represent their coordinates by three floating point numbers. The z value allows representing the room's height above ground level, so that multiple floors can be represented. It should be noticed that the z-value does not define the height of a room in terms of the distance between the floor and ceiling: this measure would rather correspond to the difference between the z values of two rooms that are situated on top of each other. Polygons can have several symbolic attributes, such as name, type, and accessibility for pedestrians. Polygons that are defined by vertices from two different levels can be used to represent connections such as ramps, stairs, or escalators.

The definition of polygons implicitly defines edges between connected vertices. Figure 70 (left) shows an example, where the polygon "*Corridor.14021*" is defined as a sequence of vertices with index (1, 2, 3, 4, 5, 6, 7, 8). This results in a set of edges, which are defined by two vertices (a, b); in our example they form the following set: {(1, 2), (2, 3), (3, 4), … , (8,1)}. In order to allow for route finding in the model, it is important to know the semantics of connections between polygons. Thus each edge is attributed by their passableness: edges that represent walls or windows are set to "*not passable*", in our example the edge (8, 1) represents a wall. The edge (6, 7) however connects the corridor polygon with the adjacent staircase polygon (7, 6, 12, 11, 10, 9). Therefore this edge is annotated to be "*passable for pedestrians*".

This example also illustrates the principle of edge-sharing between polygons. By the mutual use of vertices, all polygons form a mesh-like data structure that reflects the topology of the modeled building. This principle avoids redundancy and contradicting attributes, since edges of adjacent polygons are represented only once. It further reduces the overall number of vertices that are needed, and if applied consequently, finally leads to a partitioning of floors. Thus each point within a floor belongs to exactly one polygon, so that we have a clear mapping between coordinates and rooms that allows to make statements like "*The person Dominik is currently located in room 1.06*".

On the right hand side in Figure 70, a sample path is shown; it has been calculated by the Pathfinder algorithm based on start- and ending points within the 2.5 dimensional location model.

Now that we have explained the underlying data structure, we will introduce the user interface of our *YAMAMOTO* modeling toolkit and show how to create models in a step-by-step tutorial.

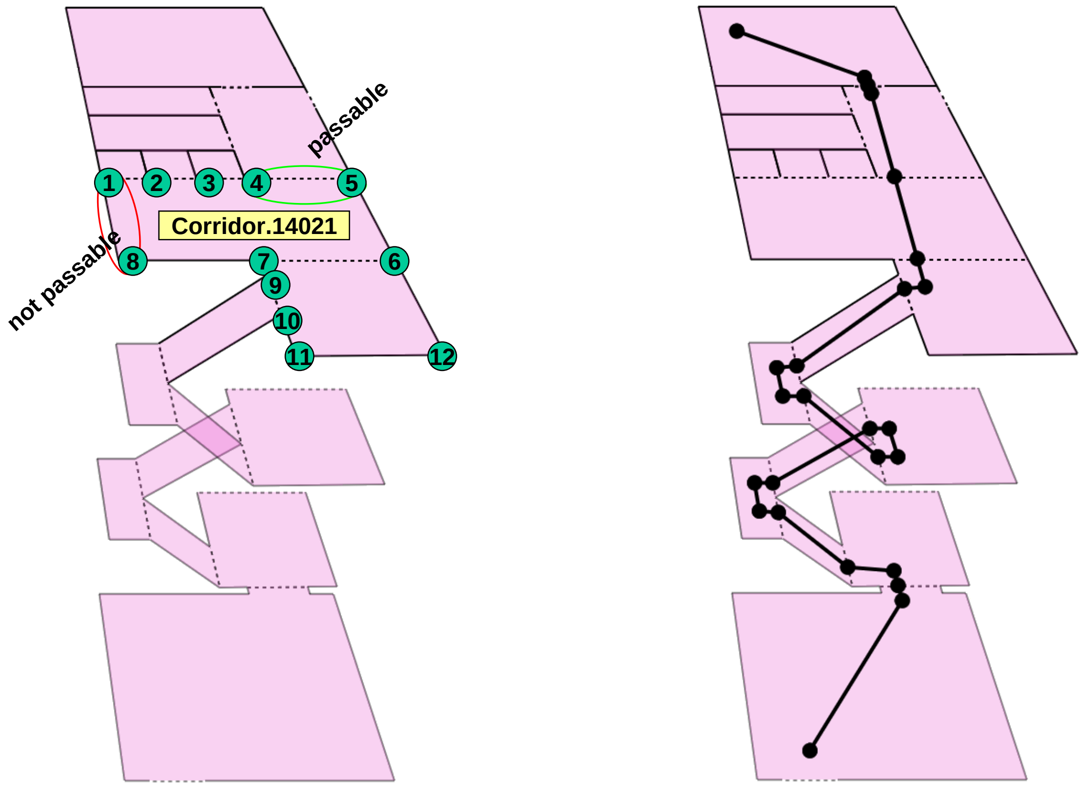


Figure 70: The 2.5 dimensional data model (left) and a route between two points (right).

7.1.1 Introducing Yamamoto

The central component of our map modeling toolkit is a graphical editor that has been implemented in the C# language on Microsoft's .NET platform. The editor allows the user to trace the silhouettes of geographical or architectural features depicted in a backdrop bitmap image and thereby to create a mesh of polygonal objects. The user interface is based on the OpenGL 3-D graphics architecture, which enables top-down projections as well as perspective views from an egocentric or allocentric perspective, free rotation and zoom. The modeling of multi-story buildings is assisted through multiple layers.

As a file format, we have chosen the XML standard to encode the model in a new human-readable file format called YML, as specified by the Document Type Definition in Appendix B, Table 25. We have written parsers for YML in the C#, C++, and Java programming languages. YML models can also be exported into the VRML (Virtual Reality Modeling Language) and RoSiML (Robot Simulation Modeling Language; see Ghazi-Zahedi et al, 2005) file formats.

Since the complexity of a model is inevitably limited by its file size, we adopt a 'divide-and-conquer' strategy. Consider for example a city model *A*, which represents the outlines of some buildings, and a refinement model *B*, which represents the rooms inside one of *A*'s buildings. The editor allows one to graphically assign *B* to its outline in *A* by direct manipulation. The link information is stored in the header of model *B*, together with the necessary transformations (translation, scale, and rotation) of *B*'s local coordinate system.

Thus higher-level models may be refined, even if the user has no permissions to modify them directly. Browsing through nested models is supported through a tree window.

As the toolkit has been designed with pedestrian navigation in mind, it includes a route finding module called Pathfinder. It is able to generate routes between any two points in a model, which follow the line-of-sight whenever possible instead of a restrictive path network. Since even multi-story buildings can be represented as a single mesh, the pedestrians will be routed through staircases, if needed. During the modeling process, the results of the route finding module can be tested at any time within the editor. For the developers of mobile applications, the routing module's *C++* code is also available.

The *GOAL* design method is also supported through the toolkit by means of an activity editor that is integrated into the user interface. The user can assign ontological concepts and instances to all geometric items, like spatial regions and furnishing objects. This extends the geometric location model to a hybrid location model (see also Section 0). Based on the hybrid model, activities can be defined according to Activity Theory and the associated artifacts and locations can be directly highlighted in the 3-D view of the environment. During the design process the integrated avatar perspective allows the designer to simulate and evaluate the instrumentation of the environment in virtual reality before the system will be deployed. Finally, the 3-D view can be utilized to monitor the real environment in the model, following the dual reality approach. For example, the avatar can be used to visualize the position of a person that is tracked in the real environment, and virtual displays can present the actual content of real displays using the VNC protocol.

7.1.1.1 *User Interface*

The layout and the components of the graphical user interface are depicted in Figure 71. On top of the screen there are the pull-down menus, and the toolbar below provides shortcuts to the most frequently used menu items via graphical icons. The current model is shown in the geometric viewing area in the center of the screen. Polygon objects can either be graphically selected in the 3-D view, or by their name from the selector list box. The model can be shown in an orthogonal projection that is similar to a map, or alternatively in a perspective view from outside (allocentric) or from inside the building through the eyes of an avatar (egocentric). Around the viewing area are scrollbars that allow to focus the view on any point in the x/y plane and to set the zoom level. On the right hand side, the top window represents the properties of the currently selected object, such as its coordinates or name. It includes an editable table that lists all attributes and their values in alphabetical order. The layer controller window in the middle has the purpose to show or hide selected levels of the building. The model tree window below shows the model hierarchy since each model can be part of or a refinement of another model. In the status bar at the bottom of the screen, the current coordinates of the mouse cursor are shown in model units and optionally also in geographic units of longitude and latitude. The UbisWorld Ontology and the associated Activity Editor are integrated by a Web-browser window to support the GOAL method.

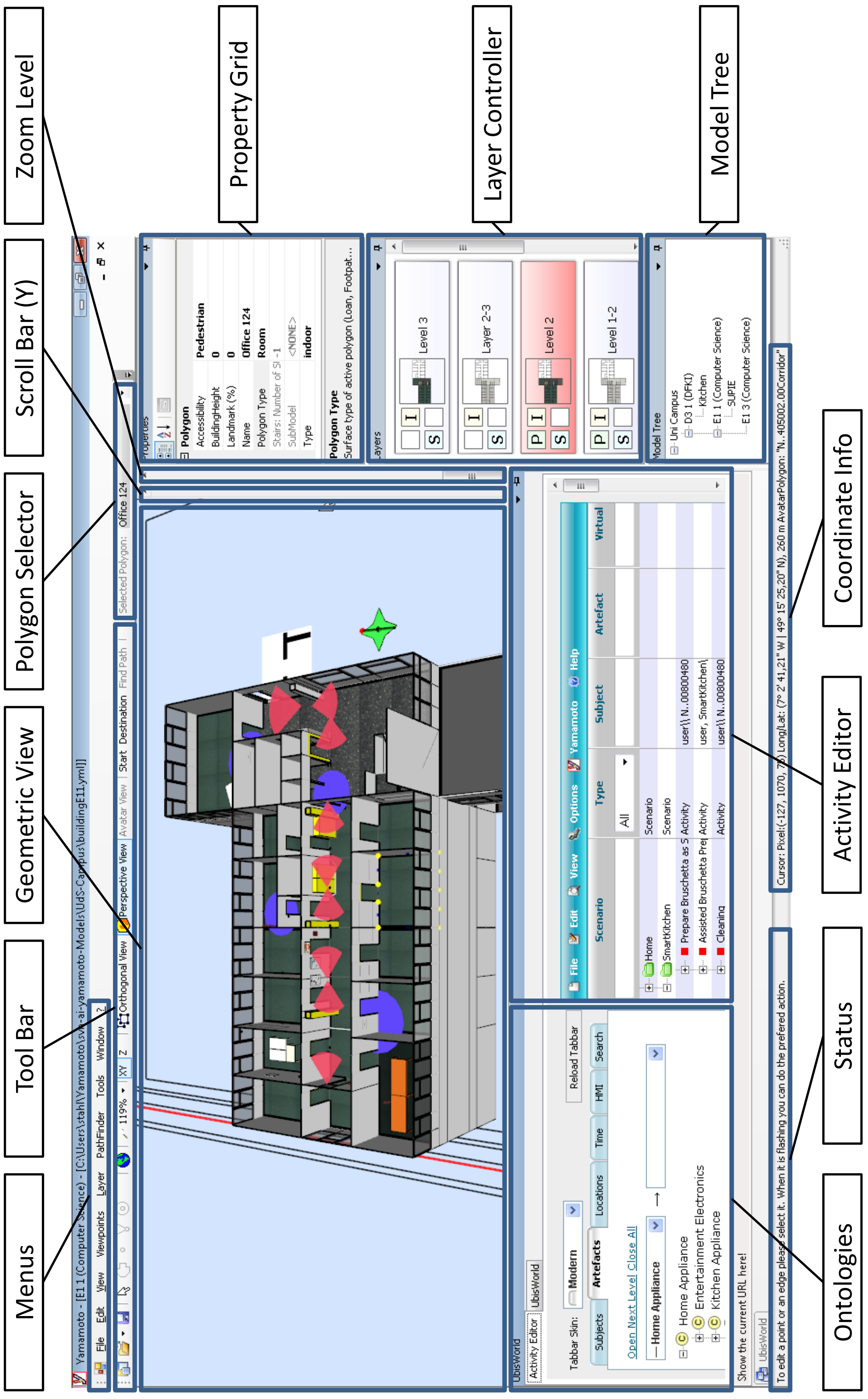


Figure 71: Elements of the Yamamoto User Interface.

7.1.2 Using the Editor

We will now describe step by step how the editor can be used to model complex environments comprising outdoor areas and the interior of multi-story buildings. The process begins with the selection of an image source that partially depicts the geographical or architectural features of the environment in an appropriate scale. In the next step, the editor allows to model the outlines of these features as a mesh of polygon objects by tracing the backdrop template. Additionally, navigational fixpoints can be marked using geometric primitives such as points or circles, and furniture items can be placed inside the rooms. If multiple levels have to be modelled, new layers should be defined. Finally, models of different scale can be linked together. The resulting models are encoded in our XML-based file format .*YML*.

7.1.2.1 *Creating a new Model*

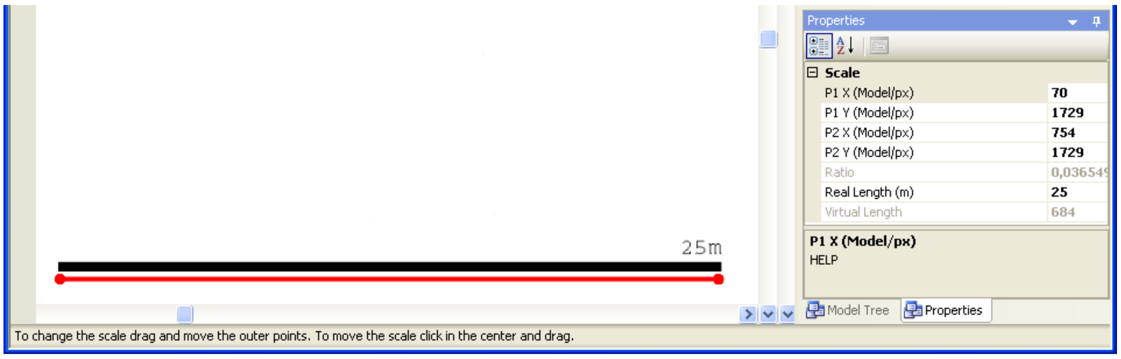
The creation of a new model begins with the selection of a backdrop bitmap image, which may be an aerial or satellite photograph for outdoor environments, as provided by companies like GeoContent GmbH or *Google Maps*[54], or an architectural plan of a building. The latter can be easily derived from CAD systems by printing views of each level into PDF file format, or by scanning existing paper plans. Architectural drawings in PDF file format are the preferred source, but they have to be exported into a bitmap file first. In order to model a multi-storey building, one bitmap image for each level is needed. A bitmap editor, such as the *GIMP*, should be used in order to crop, scale, and align the images accordingly. The suggested method is to create one project with multiple layers, and to use layer transparency of 50% to align the images using features like walls or columns that appear in multiple levels.

It should be pointed out that backdrop images should have either 512x512 or 1024x1024 or 2048x2048 pixel resolutions due to restrictions of some older OpenGL implementations. The whole project should be resized and then each layer should be saved in a separate PNG file that is named, for example, "*buildingX-levelY.png*".

The steps to create a new model and to add a scale in Yamamoto are as follows:

1. Choose "*New*" from the "*File*" menu to create a new model. The system will prompt you to select a backdrop image for the first layer of your new model. Pick the image of the first floor of your building.
2. Choose "*Add scale*" from the "*Edit*" menu to add a scale to your model, so that Yamamoto can convert model units into metric units. The scale appears on the top-edge outside of the of the backdrop image. Click on one of the endpoints to drag them into position to markup a known distance in your backdrop image. Now enter the real length of the measured distance in the scale properties "*Real length (m)*" in meter units (see Figure 72). Once the scale has been set, the coordinate calculator from the "*Tools*" menu can be used to convert model units into meter and vice versa.

[54] Google Maps Website: http://www.maps.google.com, visited 12.05.2009



Scale	
P1 X (Model/px)	70
P1 Y (Model/px)	1729
P2 X (Model/px)	754
P2 Y (Model/px)	1729
Ratio	0,036549
Real Length (m)	25
Virtual Length	684

P1 X (Model/px)
HELP

To change the scale drag and move the outer points. To move the scale click in the center and drag.

Figure 72: Setting the model scale by measuring a known distance in the floor plan.

7.1.2.2 *Viewing the Model*

The user can choose at any time among an orthogonal top-down projection (Figure 73), an isometric view (Figure 74), a perspective view (Figure 75), or an avatar view (Figure 76) by using the "*View*" menu options or the toolbar icons.

- The orthogonal view shows a top-down projection of the model. This view is recommended for creating new polygons by tracing the backdrop image.
- The isometric view shows the model in an orthogonal projection from a 60° angle in one of the four cardinal directions (North, East, South, and West).
- The perspective view shows the model from any viewpoint outside the model. This view is recommended to create stairs that connect two layers, and to get an overall impression of multi-level buildings.
- The avatar view shows the model from an egocentric perspective; the viewpoint is bound to the position and direction of the user, who is virtually represented in the model through an avatar object.

In the orthogonal and perspective views, you can pan the view by using the horizontal and vertical scrollbars around the viewing area, or you can press and hold the right mouse button to drag the model. The zoom-bar next to the vertical scrollbar allows you to magnify the view similar to a zoom lens on a camera. You can rotate the camera (viewpoint) around the focus point in the *xy*-plane in steps of 10°, as shown in Figure 77 by using the mouse-wheel or alternatively, by pressing the *PageUp* / *PageDown* keys on the keyboard.

In the perspective view, you can press and hold the Shift key in combination with the mouse wheel or *PageUp* / *PageDown* keys to rotate the camera vertically around the focus point in steps of 10°, as shown in Figure 78.

The avatar view requires an avatar object, so you possibly need to enable the avatar first by choosing the "*Enable Avatar*" option in the "*Tools*" menu and click on a position to initially place your avatar. Note that he avatar should be always placed only within polygons (for collision checking purpose). If the avatar has been misplaced, "*Enable Avatar*" can be selected again to remove the avatar, and to create a new one in a valid position. The avatar can be moved forward and back along the viewing direction by using the cursor *up* and *down* keys on the keyboard, and rotated to the left and right by using the cursor *left* and *right* keys. Holding the *Shift* key increases the speed. The avatar can also step sideways by pressing the *n* and *m* keys. The viewing direction depends on the orientation of the avatar and the position of the mouse pointer inside the viewing area, so that the avatar can look up and down. The viewpoint height depends on the "*Height*" attribute (measured in meters) in the avatar's properties, and the field of view is defined by the "*Viewing Angle*" attribute; the default angle is 45 degrees, and a smaller angle produces a zoom effect similar to a longer focal length on a camera lens. It is possible to save a snapshot of the current view at any time by choosing the "*Make Screenshot*" option from the "*Tools*" menu (supported formats are Bitmap, GIF*,* JPEG*,* PNG*,* TIFF, and EMF).

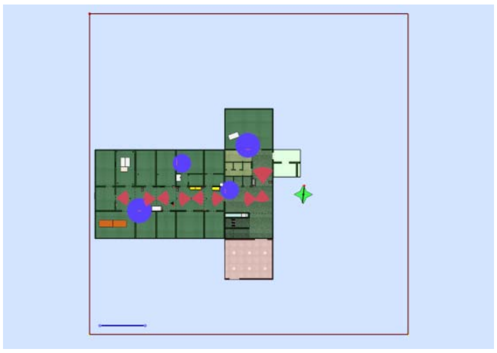


Figure 73: Orthogonal view

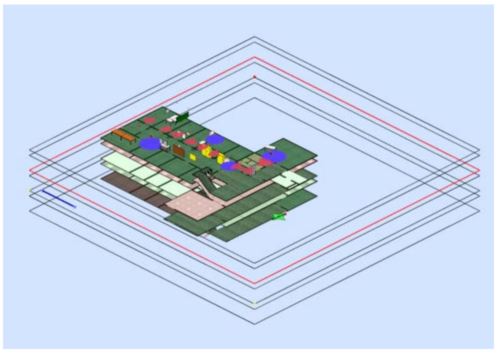


Figure 74: Isometric view

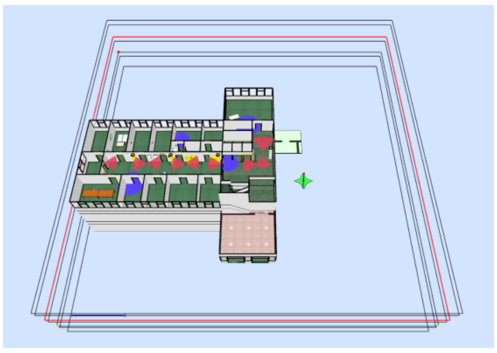


Figure 75: Perspective view

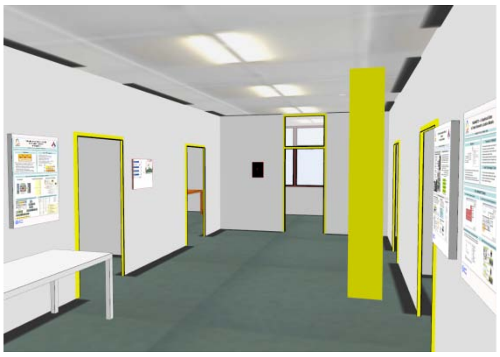


Figure 76: Avatar view

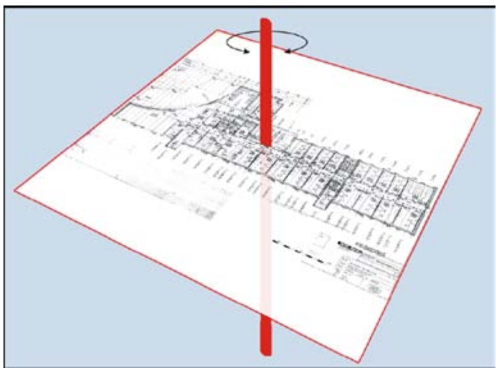


Figure 77: Camera azimuth

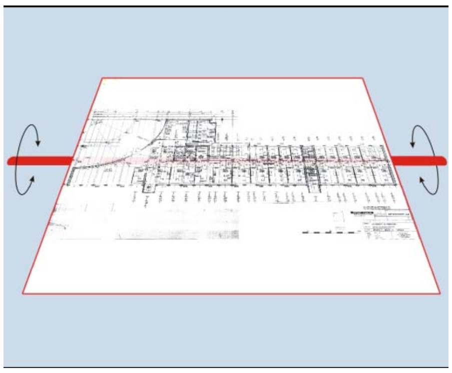


Figure 78: Camera elevation

7.1.2.3 *Modeling Spatial Regions as Polygons*

The editor has been designed to trace the silhouettes of spatial regions like geographical or architectural features depicted in the backdrop bitmap image and to model their shape through polygons. The polygons should ideally completely cover the modeled area as a mesh, but must not intersect or overlap. To achieve this, polygons should always share their vertices and edges with adjacent neighbor polygons. So before a new polygon is created, one should look ahead and think about connection points for all neighbor polygons. Additional vertices should be created where they will be needed later, typically at T-junctions between walls. We will now describe the procedure to create a new polygon:

- Choose the orthogonal view from the toolbar;
- Use the layer controller window to select the layer in which the polygon should be created. Activate the "T" (texture) option to activate the backdrop image;
- Click on the polygon creation icon in the toolbar or choose "Create Polygon" from the "Edit" menu. Deselect any active objects first by clicking on the pointer-tool in the menu bar;
- Move the mouse pointer to the first corner of the room to be modeled, and click once with the left mouse button to begin drawing (see Figure 79). Press the keyboard combination *Ctrl+C* to aboard drawing;
- Click the left button once again for each corner. Points can be removed by pressing *Ctrl+C*. Press and hold the Shift key to draw only horizontal or vertical aligned lines;
- To close the polygon, move the mouse pointer back to the starting point and the borderline color will change from blue to yellow (see Figure 80). Now click once to complete the polygon. The new polygon remains selected (see Figure 81), so that its symbolic properties can be edited.

Sometimes it is desired to copy polygons. However, the most profound difference between *YAMAMOTO* models and other 2-D or 3-D drawing software is that polygons should not be individual objects, but rather form a mesh with shared vertices and edges. This consideration has led to the decision to avoid *copy* and *paste* operations in general. The same considerations apply to the moving of polygon objects. In a mesh, moving a polygon would also deform all its neighbors and possibly twist their edges. Hence *YAMAMOTO* does not provide such a move operation. It is safer to move just single vertices.

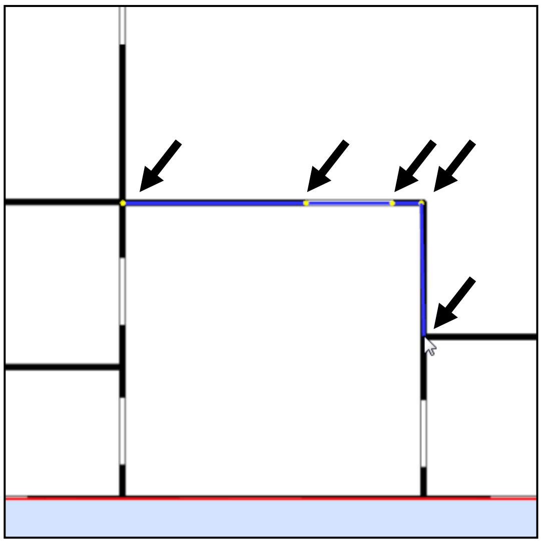


Figure 79: Creating new points (yellow) and edges (blue).

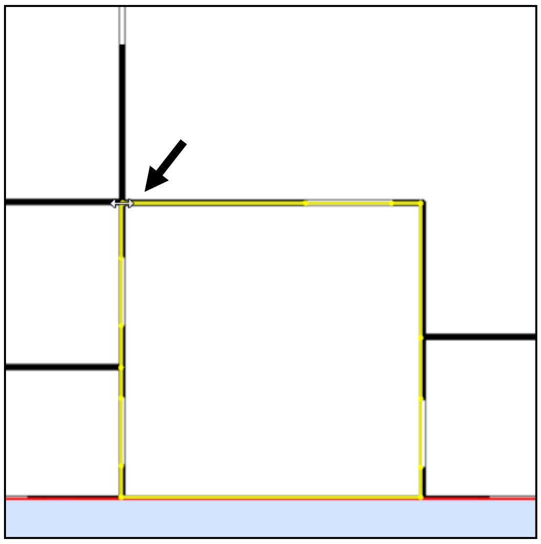


Figure 80: Selecting the starting point again allows to finish the polygon (indicated yellow).

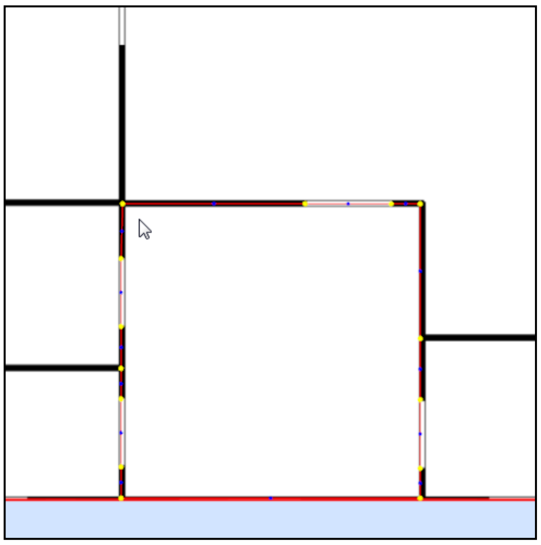


Figure 81: The finished polygon remains selected and highlighted (edges are red).

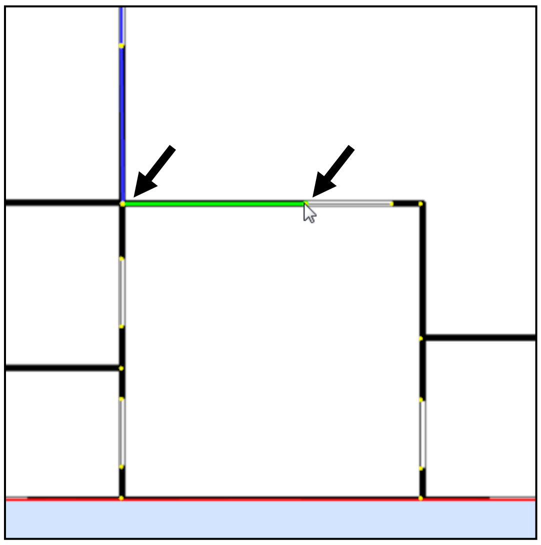


Figure 82: Connecting to existing points in the model (indicated by green edges).

7.1.2.4 *Connecting to Existing Polygons*

As said before, all polygons should be connected to their neighbor polygons by sharing their vertices. Otherwise, the route planning algorithm can't work. In order to create a new neighbor for an existing polygon (e.g. an adjacent room), the following steps are required:

- Use the layer controller window to activate the "S" (selectable) option for all layers which contain points that should be reused. It will make all points of this layer selectable;
- Click on the polygon creation icon in the toolbar or choose "Create Polygon" from the "Edit" menu. Existing points in the current layer should appear thick and yellow, and points in other (selectable) layers should be shown in blue. Any active objects have to

be deselected first by clicking on the pointer-tool in the menu bar;

- Move the mouse pointer over existing points to connect to them. The point's color will change to green and also the color of the borderline turns green (see Figure 82). If the necessary points do not exist, they have to be inserted by selecting and splitting edges first.

7.1.2.5 *Cloning Existing Points from other Layers*

YAMAMOTO supports to project existing points into other layers. Just proceed with the creation of polygons as described above in a new layer. Make another layer visible and selectable.

- It is now possible to project points from other layers into the current layer by moving the mouse pointer over them. The current edge will turn to green, as shown in Figure 83. Now by pressing the *Alt* key, the mouse cursor will appear at the same (*x, y*) position as the selected point, but on the *z* level of the current layer (see Figure 84). Clicking the left mouse button while holding down the *Alt* key creates a new point at this position. This feature is useful to "copy" some features from other layers. It is also possible to replicate the complete layer through the layer menu.

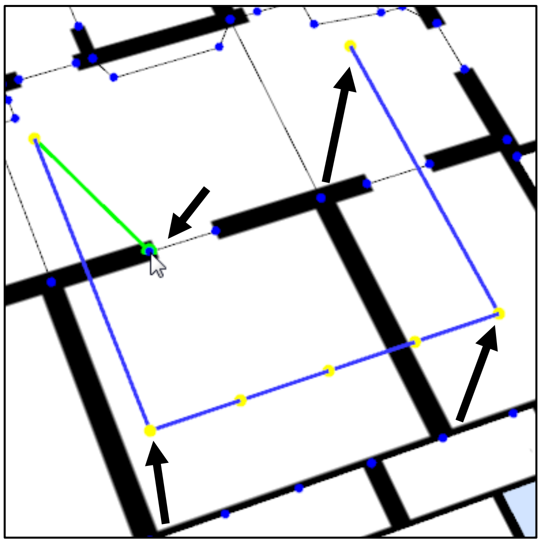


Figure 83: Move the mouse over an existing point in the layer below.

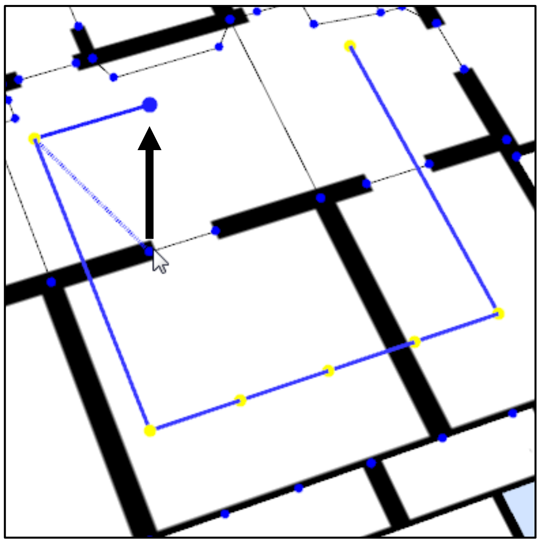


Figure 84: Press and hold the *Alt* key to project the point at the mouse in the active layer.

7.1.2.6 *Selecting and Manipulating Polygon Objects*

To select an existing polygon, make sure that the layer in which the polygon is situated is selectable and set active. Deselect and active object first by clicking into empty space or choosing the pointer-icon from the toolbar. Click inside of a polygon to select and highlight it. Alternatively any polygon can be selected by its name from the combo-box right to the toolbar. Once a polygon is selected, yellow handles allow selecting its vertices and blue handles allow picking one of its edges. Each polygon, edge, and vertex has a set of attributes which can be inspected and edited in the properties-window.

The following operations can be applied to the selected polygon:

- Inspect and modify the properties of the polygon (please refer to the Polygon Properties section).
- Move a point by selecting and either dragging it with the mouse or by editing the coordinate properties manually (please refer to the Point Properties section).
- Inspect and modify the properties of an edge by selecting it (please refer to the Edge Properties section).
- Delete a point by selecting it and choosing “Remove point” from the “Edit” menu or by pressing the key combination *Ctrl+R*.
- Insert a point by selecting and splitting an edge. Choose “Insert point” from the “Edit” menu or press the key combination *Ctrl+I*. A new point will be inserted in the middle of the edge. Known bugs: splitting edges is error prone and may corrupt its neighbor polygons. In that case, delete and re-create the corrupted polygons.
- Delete the polygon by choosing “Delete object” from the “Edit” menu or by pressing the key combination *Ctrl+D*.
- Move the polygon by choosing the “Move polygon” option from the “Edit” menu, which moves all its vertices by a given offset along the *z* axis.
- Change the polygons parent layer by choosing “Move polygon to layer” from its context menu via the right mouse button. This will leave the geometric position of the polygon unchanged, but moves it within the logical model hierarchy of layers.

7.1.2.7 *Object Properties*

The *Properties* window allows the user to inspect and manually edit the attributes of polygons, edges, and points. If a polygon is selected, its name (eventually an UbisName that refers to a location in the *UbisWorld* Ontology), type (room, stair, elevator, building), and accessibility (for pedestrians and/or cars) are displayed and can be changed. The polygon type has influence on the rendering style and navigation; stairs are displayed with multiple steps, and buildings are displayed with walls and a flat roof. The elevator type signals the route finding algorithm that marked platforms with the same name are interconnected and provide a shortcut between levels. See Table 18 in Appendix A for a detailed reference of all polygon attributes. By clicking on a polygon’s vertex, the geometric attributes of the point are displayed and can be manually corrected, if necessary. Please see Table 19 for all point attributes. Likewise, edges can be selected in order to set their passableness and visual rendering style (open, wall, window, etc.), see Table 20. Please note that shared edges between two adjacent polygons with the same vertices have the same properties.

7.1.2.8 *Working With Layers to Model Buildings with Multiple Floors*

Buildings with multiple stories can be modeled efficiently through the concept of layers. These layers define new workspaces above the ground plane, and use their own backdrop images. One can think of layers as a vertical stack of floor plans. The intention of layers is to simplify the creation of new polygons in 3-D space and to group all polygons of a floor into a single object. All layers can be selectively shown or hidden to ease the handling of complex buildings. To avoid confusion, all polygon operations are restricted to a single ‘active’ layer. Polygons can only be selected from the active layer, and new polygons are always created on the *z* level of the active layer.

The layer controller window represents all existing layers of a model in a top-down ordering of layer control elements (see Figure 85). The active layer is highlighted in red color. From left to right, each layer control element contains checkboxes, an iconic representation of the backdrop image, and the name of the layer.

The "P" checkbox in the upper left corner is by default active and switches all polygons belonging to this layer on or off. Turning polygons off helps to un-clutter the viewing area, but it can also be used to find out about the structure of a model. Likewise, the "I" checkbox switches all geometric items on or off (these are points, sectors, and spheres that represent input and output devices in instrumented environments). The "S" checkbox makes all polygons of the layer selectable. During the creation of new polygons, it determines whether the points of this layer can be connected to and re-used. The "T" checkbox fills the whole layer with the backdrop image texture, if enabled. This setting is recommended to create new polygons. If the "T" symbol is disabled, the backdrop image is projected as a texture only on the modeled polygons, and other areas appear empty.

Independently of the texture checkbox setting, the backdrop image can be rendered in three different modes: disabled, transparent (50%), and opaque. The mode can be changed by clicking on the icon. If the image is disabled, a cross appears instead of the icon.

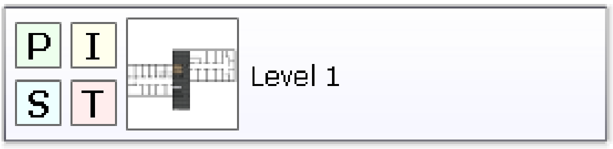


Figure 85: Each layer control element contains checkboxes, an iconic representation, and name.

It is possible to modify, add, duplicate, and remove layers at any time by pressing the right mouse button in the layer controller window to open the layer context menu (Figure 86).

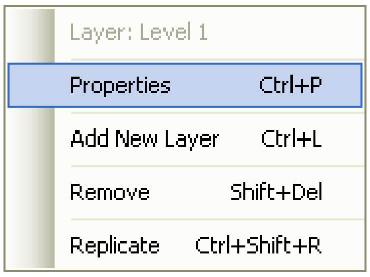


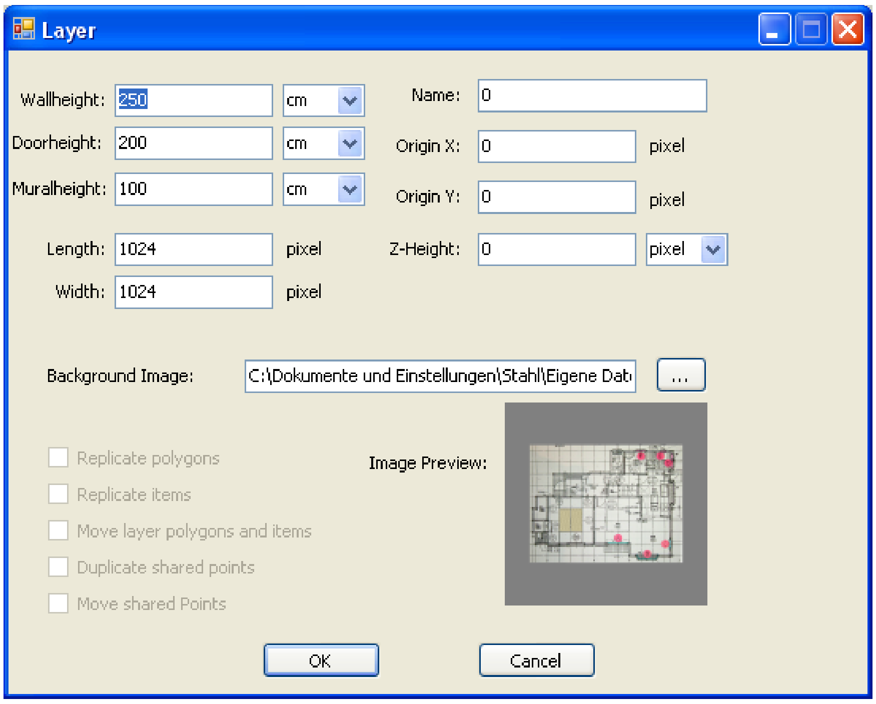
Figure 86: Layer controller context menu.

In the following we explain how to use the layer controller context menu (Figure 86):

- **Layer Properties.** The first entry in the context menu opens the layer properties dialogue window, as shown in Figure 87. The layer attributes comprise name, coordinate origin, and background image of the layer. For the 3-D visualization, the default height of walls also needs to be defined. For a detailed description of all layer attributes, please refer to Table 21 in the appendix.
- **Moving a layer along the z axis.** Changing the "Z-Height" value of a layer moves the origin along the *z* axis and enables the checkbox "Move layer polygons and items". By checking this option, all polygons (and items) of the layer will be moved to the new

position. However, the principle of meshed polygons leads to ambiguous results if polygons which connect two layers are moved. Hence the user can decide if connected points in other layers should be also moved by checking the "Move shared points" option. Alternatively, such polygons can be disconnected from the other layers by duplicating their common points.

- **Adding, Removing and Replicating Layers.** It is possible to add and remove layers at any time during the modeling process, as necessary. If you add a new layer you have to choose a backdrop image and enter its Z origin (height above ground). Replicating a layer means to create an exact copy at a different height. The new layer will have the same properties, and it will include copies of all polygons if the "Replicate polygons" option is checked.


Layer

Wallheight: 250 cm
Doorheight: 200 cm
Muralheight: 100 cm
Length: 1024 pixel
Width: 1024 pixel
Name: 0
Origin X: 0 pixel
Origin Y: 0 pixel
Z-Height: 0 pixel
Background Image: C:\Dokumente und Einstellungen\Stahl\Eigene Dat
Image Preview:

- [ ] Replicate polygons
- [ ] Replicate items
- [ ] Move layer polygons and items
- [ ] Duplicate shared points
- [ ] Move shared Points

Figure 87: Layer properties dialog box.

7.1.2.9 *Creating Stairs between two Layers*

One of the most important features of Yamamoto is its ability to represent staircases between the floors of a building. We have already explained how a mesh of polygons is created by re-using vertices, so the following steps should be relatively easy to follow. We will begin with a simple stair that directly connects two layers before we explain how to construct a staircase with platform between two layers.

Creating a simple stair:

1. Create edges to which the new stair should connect to in both layers (either create new polygons or split existing edges);
2. Make both edges passable;
3. For both layers, activate the "S" (selectable) option, activate the "P" (polygons) and deactivate the "T" (texture) option;

4. Activate the layer to which the stair object should belong to (its parent layer). Each object (polygon, item, display, or box) internally belongs to exactly one layer. The polygon's context menu can be used to change its parent layer any time;
5. Zoom and rotate the view (using the mouse wheel, the shift key, and zoom bar) so that both edges and their vertices can be seen (two yellow points in the active layer, two blue points in the other layer you want to connect);
6. Click on the "*Create Polygon*" tool or choose "*Create Polygon*" from the "*Edit*" menu;
7. Move the mouse pointer over one of the edge's points in the active layer. The points color should change from yellow to green. Now click (left mouse button) to start creating a polygon beginning from that point. Now move the mouse to the other point and click again;
8. Move the mouse to the blue point (on the same side) in the other layer. It should also become green and then click again. Repeat for the other blue point. Finally, click on the point you started with to finish the polygon (the line's color will change to yellow);
9. In the properties panel, set the polygon type to "*Stairs*". The rendering style will change and show steps and handrails. The number of steps and the type of handrail (open or mural) can be changed as required.

7.1.2.10 *Creating Stairs with a Platform between two Layers*

In order to create a stair between two layers, the following steps are required:

1. Create a platform polygon with at least 5 points (5 edges). Make two of them passable;
2. Vertically move the platform between both layers: Choose "*Move polygon*" from the "*Edit*" menu and enter the height difference. Alternatively, one can manually select each point and edit its *z* coordinate in the properties panel;
3. Continue with the procedure for creating a simple stair between the lower level and the platform, and repeat it once for the upper stair.

7.1.3 3-D Modeling with Paramentric Objects

In this section, we describe how a 2.5-dimensional model can be easily extended to a full 3-D model that is suitable for visualization from an egocentric perspective. Typical 3-D modeling tools use cuboids for walls and cut-out objects for doors and windows. These tools are complex to use and difficult to learn. Our approach strives to minimize the effort that is required for the model. The *YAMAMOTO* editor semi-automatically creates the building structure from the 2.5-dimensional polygons and the interior can be easily modeled through box-shaped primitives. Regarding the structure, one can quickly annotate each existing edge by selecting their desired appearance from a choice of predefined types. Edges can be rendered as walls, doors, walls with window, mural, handrail, or just open space. The corresponding geometries are created "on-the-fly" in the perspective and avatar views. Our approach presents an abstraction from reality as it represents walls as flat surfaces without thickness and hence trades in realism for the sake of simplicity.

The egocentric viewpoint from the inside of a building also creates a demand for furniture items and appliances. Such objects are also very important to model typical activities in the environment. For example, the design of a user assistance system for a kitchen environment requires detailed knowledge about the position and arrangement of the shelves, the cooker, refrigerator, and so on. Hence in *YAMAMOTO* rooms can be equipped with simple 3-D objects that are all derived from a box primitive described by width, depth, height, and orientation. Depending on its type, the box will render as shelf, table, seat, or home appliance. The latter uses image textures of the original appliance front, such as a TV set or a dishwasher, for an almost realistic appearance without the need for additional complex geometry.

We will now introduce the building structure annotation in depth, followed by the interior object properties.

7.1.3.1 *The Building Structure*

The *YAMAMOTO* modeling toolkit provides a kind of construction kit that includes basic geometric structures for walls, doors and windows. The rendering of these elements can be selected by choosing the "*Show Walls*" option from the "*View*" menu. Window panes can be visualized with a glass-like transparency effect by choosing the "*Show Semi-Transparent Windows*" option from the "*View*" menu.

The set of elements is shown in Figure 88; from left to right: "Wall", "Door", "Window", "Window with skylight", "Mural", and "Handrail". Additionally, a "Door with skylight" and a "Painted wall" element can be chosen. The latter has the same geometry like the normal wall, but it has a user defined color instead of the white default color.

The vertical dimensions of the elements are defined by the three base-lines called "Mural Height", "Door Height" and "Wall Height". Their *z* values are set in the layer properties dialog. It is notable that all *z* values are specified in centimeters, so that a valid scale element is required in the model. The layer properties are only default values and they can be individually overridden for each edge. Furthermore, it is possible to customize the color and width of door- and window-frames. For a reference of all attributes see Table 20.

For the convenient editing of multiple elements, such as a building front, all properties can be copied to the clipboard by choosing "*Copy Properties*" from the edge's context menu. By selecting another edge and choosing "*Paste Properties*", all values (type, heights, and color) are automatically transferred and set.

Figure 89 shows the previous example from Section 7.1.2.3 again and how the 3-D geometry is created step-by-step (from left to right) by defining edge attributes.

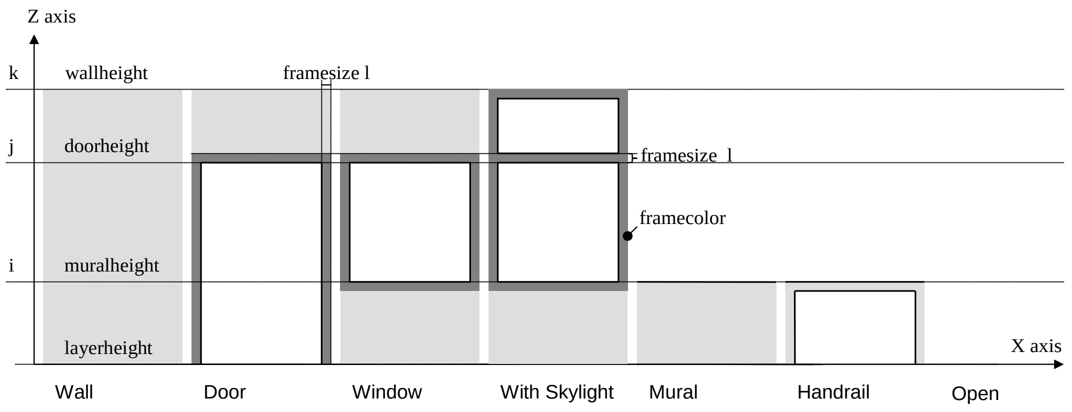


Figure 88: A set of building blocks allows the user to semi-automatically create 3-D geometry from 2-D data. Some attributes can be used to customize the elements.

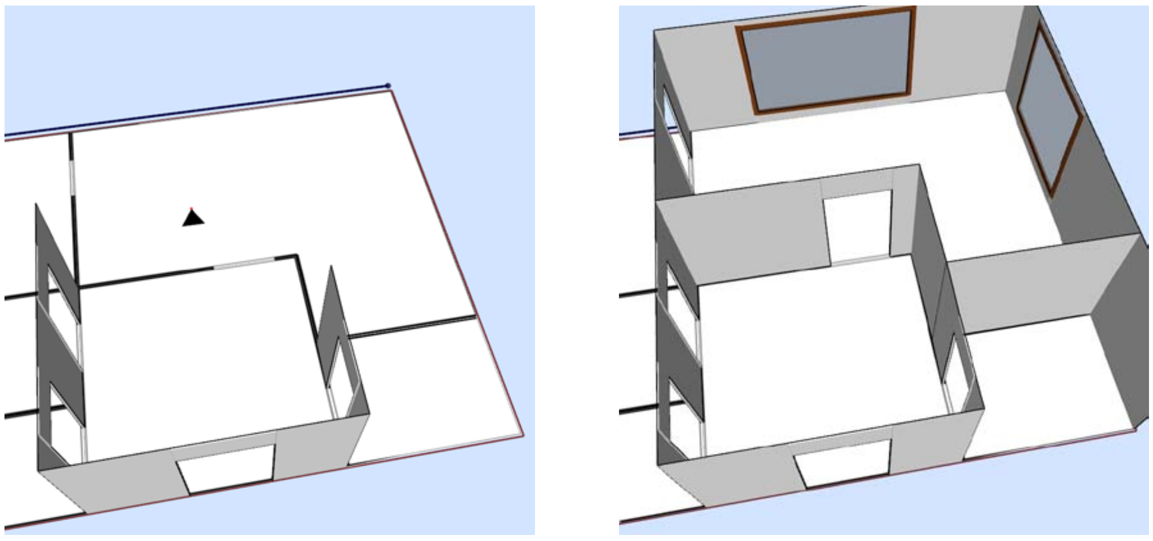


Figure 89: Defining the edge attributes creates walls, doors, and windows in the example model.

7.1.3.2 *Furnishing of the Interior*

The visualization of the room's interior is important for two practical applications of the *YAMAMOTO* toolkit, namely the visualization of route descriptions and the modeling and simulation of interaction in instrumented environments. Good route descriptions include references to salient objects along the way, so called landmarks. Outdoors, noticeable buildings or towers can be landmarks. Indoors, one might refer to furnishing items like counters or paintings. The task of activity modeling goes even deeper and requires all displays, storages, appliances, and artifacts to be properly represented. Since our design goal is to keep the modeling as simple as possible, we have implemented a set of frequently-used objects. Most appliances tend to have a rectangular shape and only differ in their size and front. So we have implemented a parametric 'box' object in *YAMAMOTO* that has the basic properties of position, width, depth, height, and orientation as shown in Figure 90. Besides choosing from a set of plain colors, image textures can be mapped to the front and top surfaces of the box in order to give the box a more realistic visual look, e.g. like a dish-washer. For other common object types we have implemented generic geometries that can be chosen by the "Type" property. Some of them are shown in Figure 91. The "Table" geometry is made of a surface with four legs. In order to model storage systems like shelves, the "Storage" element represents the corpus of a shelf or cabinet. Both types can be customized by the "Thickness" attribute; the width of the table's legs can be set as well as the thickness of the shelf's boards. For an overview of all properties, please refer to Table 22.

Finally, parametric box objects can be grouped together to form complex objects, such as a shelf or kitchen block. Yet the hierarchic modeling is currently not well supported in the graphical user interface. Instead we recommend writing the necessary model code in a XML editor and open it in *YAMAMOTO* to view the results, similarly to manual *HTML* design. The <box> elements can be nested, i.e. one box element can appear inside of another. It is important to know that the nested box element inherits the position and orientation of its parent box element. A stack of three boxes for example would be best modeled as four box elements: one "Group" type box that defines the position, and three children "Box" elements that only need to specify their size and z position above ground. The "Group" type omits any geometry and is only used to group its children box elements. The position and dimension attributes can be specified either in model units (pixels) or meters by adding a unit to the decimal value. For example *length="100"* means that the box has a length of 100 model units, whereas *length="100 cm"* specifies the length as one meter and is the same as *length="1 m"*.

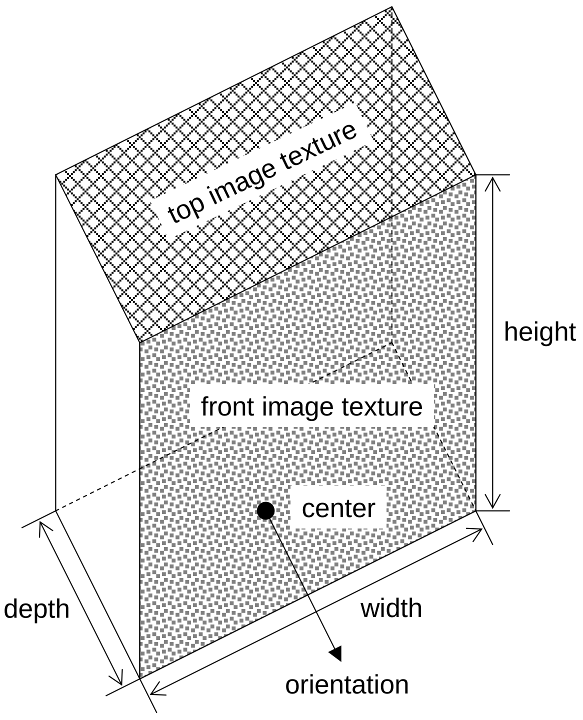


Figure 90: The box element and its basic geometric properties.

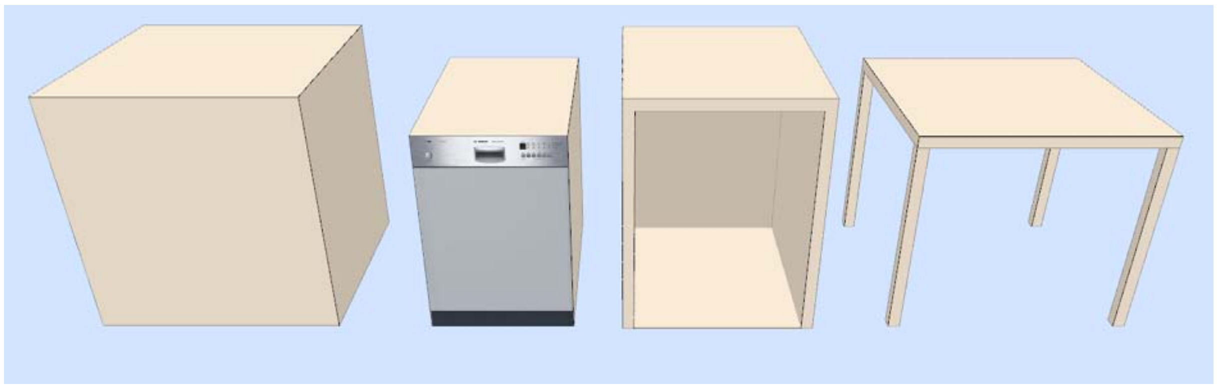


Figure 91: *YAMAMOTO* provides different generic geometries, from left to right: box, box with dishwasher-like front texture mapping, storage, and table.

7.1.3.3 *Importing Polygon Models from RoSiML*

In case that organic shapes have to be modeled, or even if just a higher level of detail is desired, alternative solutions are required. We have therefore implemented an import module that allows to use geometry from *SimRobot* scenes, which is defined in macros.

- The RoSiML (**Ro**bot **Si**mulation **M**odeling **L**anguage; see Ghazi-Zahedi et al., 2005) macros first have to be included manually as a resource to the YML model file; this is not yet supported through the graphical user interface. At the beginning of the XML file, a <Resource> element has to be added that includes one or more <RoSiInclude file="filename.rsi"/> elements that refer to RoSiML include files that contain macros. Currently, the import module only handles *ComplexShape*, *Box*, *Hinge*, and *Slider* elements and their *Translation* und *Rotation* nodes. Rotation is applied first.
- Now the included macros can be added to the YAMAMOTO model by choosing *„Add RoSiML Macro as Box"* from the *"Edit"* menu. The available macros are listed in a submenu, and can be added to the model. They are realized as yet another type of parametric object. They can be positioned and rotated, but have no additional parameters.
- Each macro should represent exactly one geometric object. In case that objects are constructed from many macros, their geometry should be merged.

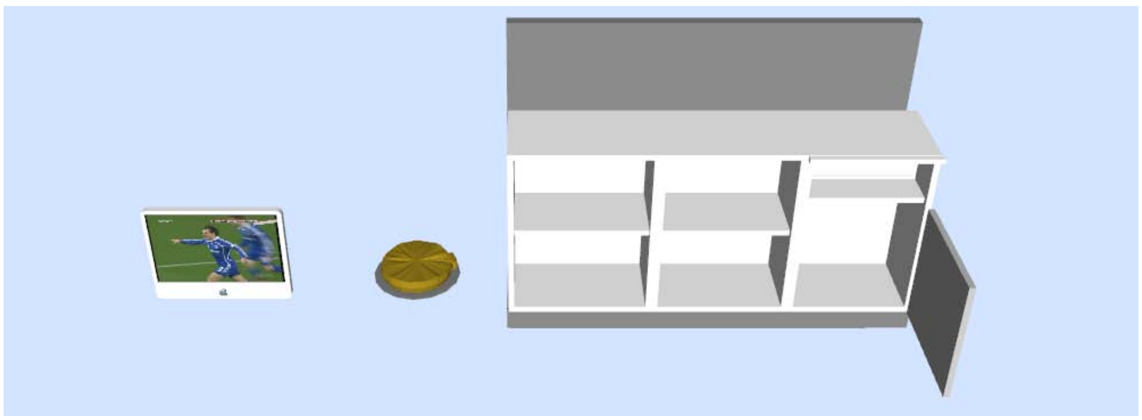


Figure 92: RoSiML example macros (from left to right): iMac, cheesecake, and kitchenette.

7.1.4 Modeling Sensors and Actuators in Instrumented Environments

One of the main goals of the *YAMAMOTO* toolkit is to support the development process of instrumented environments that comprise various sensors and actuators. The sensors are required to sense the context of the users, mainly their position. Actuators, such as public displays or the steerable projector from the SUPIE environment, are needed to convey useful information. Actuators are also required to manipulate the physical environment on behalf of the user. In case of BAALL, the kitchenette (see Figure 92) is such an actuator, since its height can be adjusted by the assistance system to suit its user. According to the *GOAL* design method (see Chapter 6) *YAMAMOTO* has been designed to assist the designer in the beginning of a project to plan the instrumentation and to decide where the sensors and actuators should be placed. The geometric information can be used later for the implementation of the system. For example, in order to build an indoor positioning system, it is important to know the coordinates of navigational beacons in a building. Finally, the model of the instrumented environment can be used for simulation, evaluation, and monitoring of the environment. In this section, we will first describe how the beacons of an indoor positioning system can be modeled and then we will show how to model public displays in the environment. We will show later in Section 7.4 how the displays and positioning engine can be simulated or connected to external software for monitoring.

7.1.4.1 *Beacons for Indoor Positioning*

Indoor positioning systems require specialized hardware since the GPS satellite signals have a weak signal in buildings and suffer from multipath problems that are caused by reflections on walls. The position of the user is usually determined by trilateration techniques, where the distances between a single mobile positioning terminal and multiple fixed sender (beacons/actuators) or receiver units (sensors) is measured either directly, based on signal runtime, or indirectly, based on the signal strength. Hence it is important to know the precise position of the beacons or receivers. Many different transmission techniques can be employed; radio communication signals (e.g. WLAN hotspots, Bluetooth Access Points, RFID) are typically omnidirectional, whereas ultrasonic or infrared signals emit a directional beam.

In order to model the different signal characteristics, the Yamamoto editor supports three basic primitives: i) *Point (x, y)*; ii) *Circle (x, y, radius)*; and iii) *Section (x, y, radius, beam angle, orientation)*. They can be given a symbolic name and an additional ID that is broadcasted by the beacon. *Point* primitives for example can be used to map the position and MAC-address of WLAN access points without considering their range. Short-range signals can be modelled as *circle* primitives with an approximated reception range. Directional signals, like infrared lights, can be defined as *section* primitives with angle and orientation.

Point primitives can be added to the model by choosing *"Add Item-Point"* from the *"Edit"* menu and pointing and clicking to the desired position in the model. Now the point appears as black dot, and a name and info can be given to it in the properties window. The semantics of the point can be selected from a list of devices, such as *rfbeacon*, *rfreader*, *microphone*, etc. (see Table 23 for more details on these properties). *Sector* and *Circle* primitives can be added likewise, but they provide an additional control point to interactively define their range and angle, respectively.

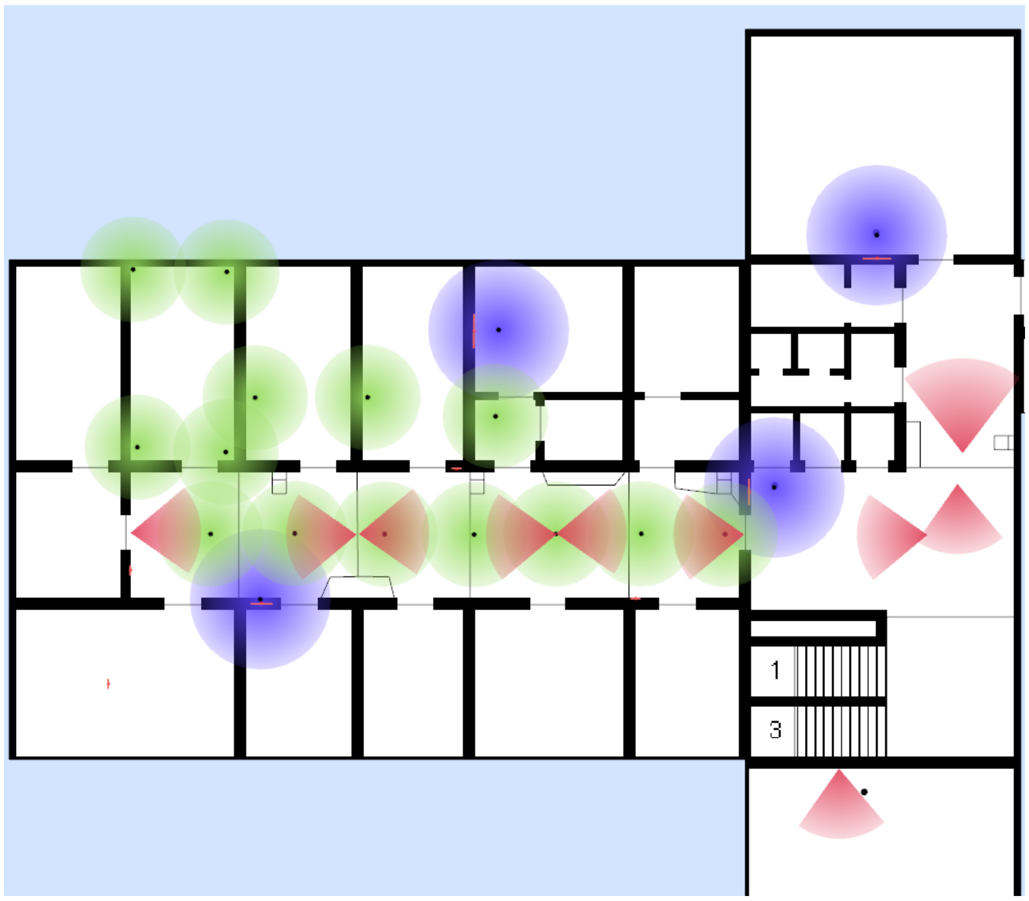


Figure 93: Infrared (red), RFID (green), and Bluetooth beacons (blue) for indoor positioning.

Figure 93 depicts the indoor-positioning beacons that we use in our lab for *LORIOT*: i) infrared beacons, shown in red, are modeled as sections; ii) active RFID tags, shown in green, are approximated as circles with limited reception range; and iii) Bluetooth dongles are shown in blue. *LORIOT* has been designed for seamless indoor and outdoor navigation (see also Section 7.6), so that it is important to use a unified, GPS-compatible coordinate system. By geo-referencing the building model, it is possible to automatically derive Longitude and Latitude coordinates for each beacon and store the coordinates into the internal memory of the RFID tags. Hence the beacons provide position signals that can be mixed with satellite data and allow for a seamless transition between outdoor and indoor environments.

7.1.4.2 *Display Objects*

The importance of displays for instrumented environments has already been discussed in Section 6.3.6.1. *YAMAMOTO* accommodates the modeling of displays with a dedicated object type that supports a static image texture, as shown in Figure 94, but also implements a VNC client to capture and map content from real displays (VNC server) to the virtual screen. The size and format of the display object can be defined to match any real display regarding its diagonal measure, aspect ratio (4:3 or 16:9), and portrait or landscape orientation. The display can also be rotated vertically, so that tabletop displays as well as projections to the ceiling of a room can be modeled. Any screenshot can be mapped to the display as static image texture. For a complete specification of the properties, please refer to Table 24. We will explain how to use the VNC client later in Section 7.4.3.

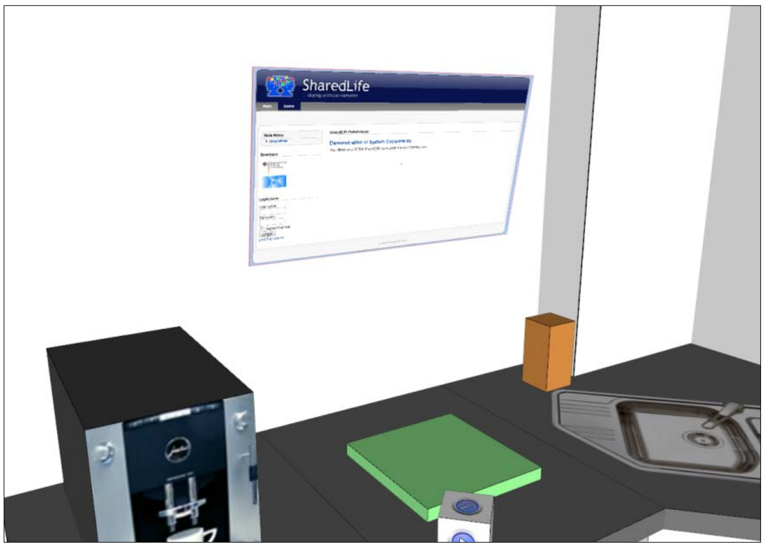


Figure 94: Example of a display object in the *Smart Kitchen* environment (see 8.1).

7.1.4.3 *Projectors and Cameras*

The placement of projectors is even more interesting and challenging than displays, because the designer has to consider the relation between the distance from the canvas and the resulting diameter of the projected image. Even though most projectors are equipped with a zoom lens that allows the user to adapt the image size to the distance, the optics has a limited range. In case of the steerable projector from the SUPIE environment the question is which areas are suitable for projection, and which objects will cast a shadow. Hence we have implemented a projector object to simulate these effects through a visualization of the viewing pyramid, as shown in Figure 95. This model helps to assess possible configurations considering the room's furnishing and the optical limitations of the device. However, the current implementation only serves as a proof of concept and does not implement a correct shadow model yet; this remains work in progress.

Likewise, the mounting of cameras poses almost the same problem as the projector, since it is difficult to estimate the field of view. Since the parameters of the optical lens are very similar in terms of focal length and viewing pyramid, the projector model might also serve as a camera model to find the best location for a camera.

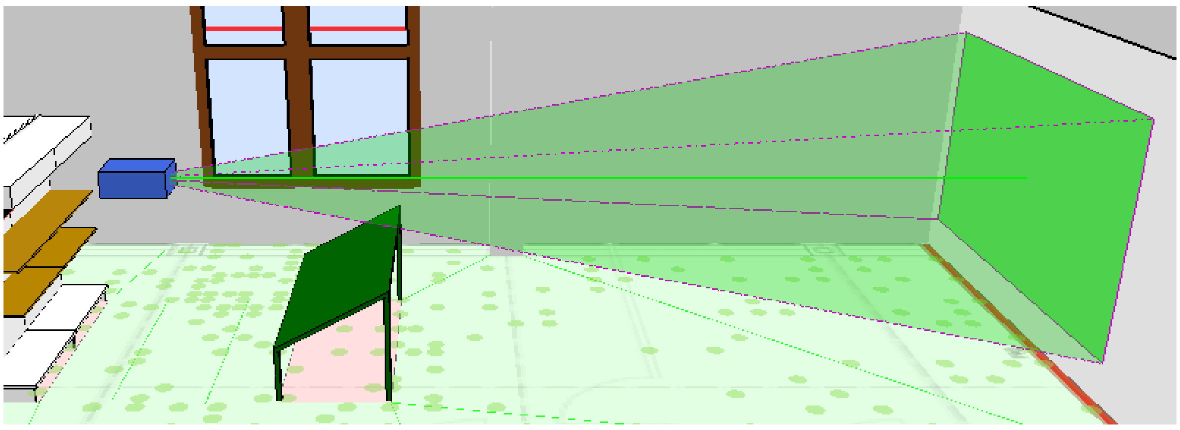


Figure 95: Modeling a steerable projector and its viewing pyramid.

7.2 Hybrid Location Modeling

We have seen how to model the geometry of the building and its furnishing in terms of position, size, and shape. What matters though is how other important properties of the building can be represented, such as organizational units, opening hours, and access restrictions. Considering artifacts, information about the vendor and technical specifications are also of interest and should be modeled.

We have decided to separate the geometry from other information as far as possible by using two different models, a geometric model and a symbolic model. Together they form a hybrid model as described in Section 2.4.5. We will now show how the geometric *YAMAMOTO* model can be extended to such a hybrid location model with the help of the *UbisWorld* ontology. All objects in *YAMAMOTO* models can be named, and by using the unique element identifiers from *UbisWorld* (so-called UbisPointers) as names, we have a bidirectional mapping between both models. By doing so, we can utilize *UbisWorld* to represent relationships between elements, for example grouping rooms into logical units like workgroups, and make statements about elements. These statements can express the current state of an element, for example where it is located, or properties, such as the temperature inside a room.

In the following we will first show how to assign elements of *UbisWorld* to geometric objects in *YAMAMOTO*. Then we will describe how to lookup elements in *UbisWorld* in order to inspect their properties and statements about them.

7.2.1 Assigning UbisWorld Elements to Yamamoto Objects

Besides the 3-D viewing window of the geometry, the *YAMAMOTO* user interface also provides a customized view on the Web-based *UbisWorld* interface. This window normally resides folded at the bottom of the *YAMAMOTO* user interface. It can be opened by clicking on the tab bar, and the user has to log into the system first with a username and password, or as a guest user. The *UbisWorld* window is divided into two frames, on the left hand side are five tabs that contain different parts on the ontology for the purpose of activity modeling (subjects, artifacts, locations, time, and human-environment interaction).

In order to assign an *UbisWorld* concept (or instance thereof) to a *YAMAMOTO* object, an element has to be selected first from one of the five tabs. In the next step, a polygon object has to be selected in the 3-D view. Right-clicking on it opens the context menu, where the user should choose the "Assign <*UbisName*>" option to assign the unique identifier of the chosen *UbisWorld* element to this object, essentially by renaming it (Figure 96).

Now the properties window will show the label-part of the assigned spatial element, which appears write-protected, since the label of the spatial element should only be changed in the ontology in order to avoid inconsistency. If the association with the spatial element should be deleted, choose the *"Remove UbisName"* option from the context menu and the name property will be cleared and unlocked.

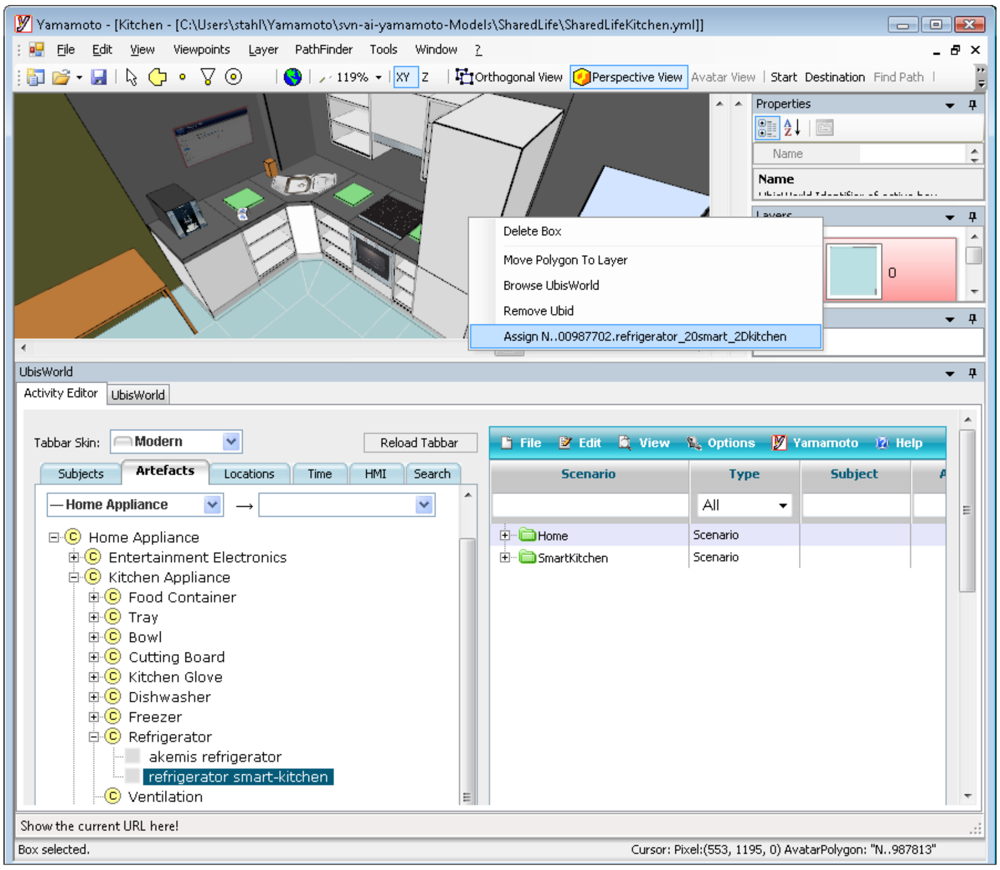


Figure 96: Assigning a symbolic element from *UbisWorld* to the refrigerator box object.

It is also possible to assign elements from the physical ontology to box-type objects in the Yamamoto model. The procedure follows the same steps mentioned above for polygons. First select an element from the artifacts or human-environment interaction tab, then right-click on a box to open the context menu and choose *“Assign”*.

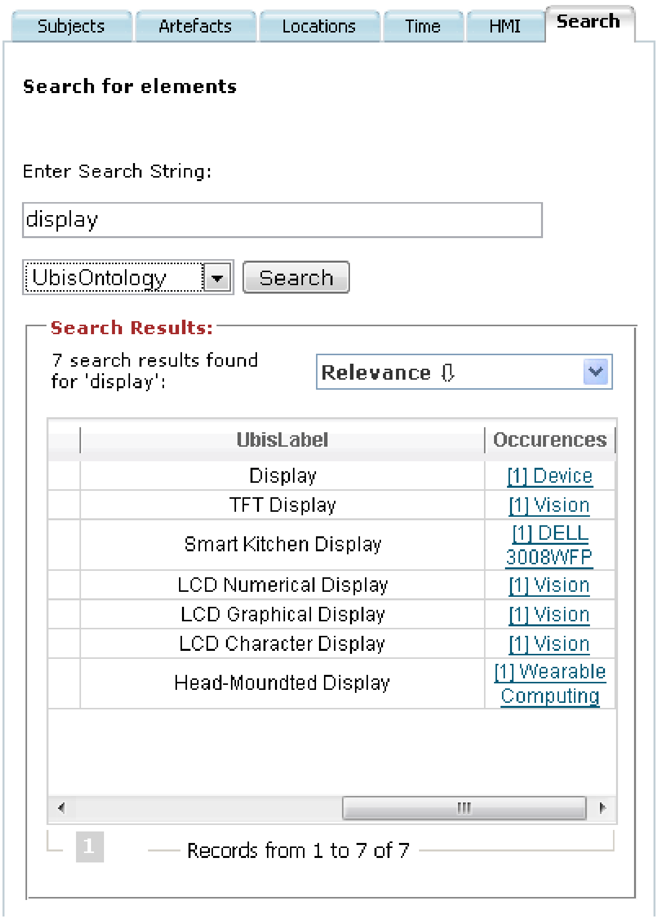
It is worthwhile to mention that Yamamoto does not validate the plausibility of the assignment made by the user; this means that the user can assign any concept or element to any polygon or box object.

7.2.2 Browsing Elements in the UbisWorld

Our motivation for the construction of a hybrid model is to link additional information to *YAMAMOTO* objects. Now that we have seen how to create a hybrid model, one might ask how this information can be retrieved from *UbisWorld*. Besides, the tree view of the *UbisWorld* interface also implements an element view that lists all statements that are made about an object. It is possible to show this element view in the browser by selecting a *YAMAMOTO* object that has an UbisName associated with it and right-clicking on it opens the context menu. By choosing *“Browse UbisWorld”*, the element view will be loaded in the “UbisWorld” tab of the browser window.

7.2.3 Searching Elements in the UbisWorld

UbisWorld provides a search function to lookup classes and instances by their name. The search string can be entered in the Search tab of the Activity Editor. The system can also be restricted to a certain branch of the ontology, such as geographic names, using the selection box. Once the search has been started, the results will be shown in the frame below, ordered by their relevance. Figure 97 shows an example, where the ontology has been searched for elements that include the word "*display*" in their label. Since *UbisWorld* allows that elements can have multiple roles, they can occur in different sub-trees. All occurrences are listed. If one occurrence is selected, it will be loaded and selected in the appropriate tree tabs, according to the element's role in the activity model. The tree tabs will be explained in more detail in the next section.



7 search results found for 'display':

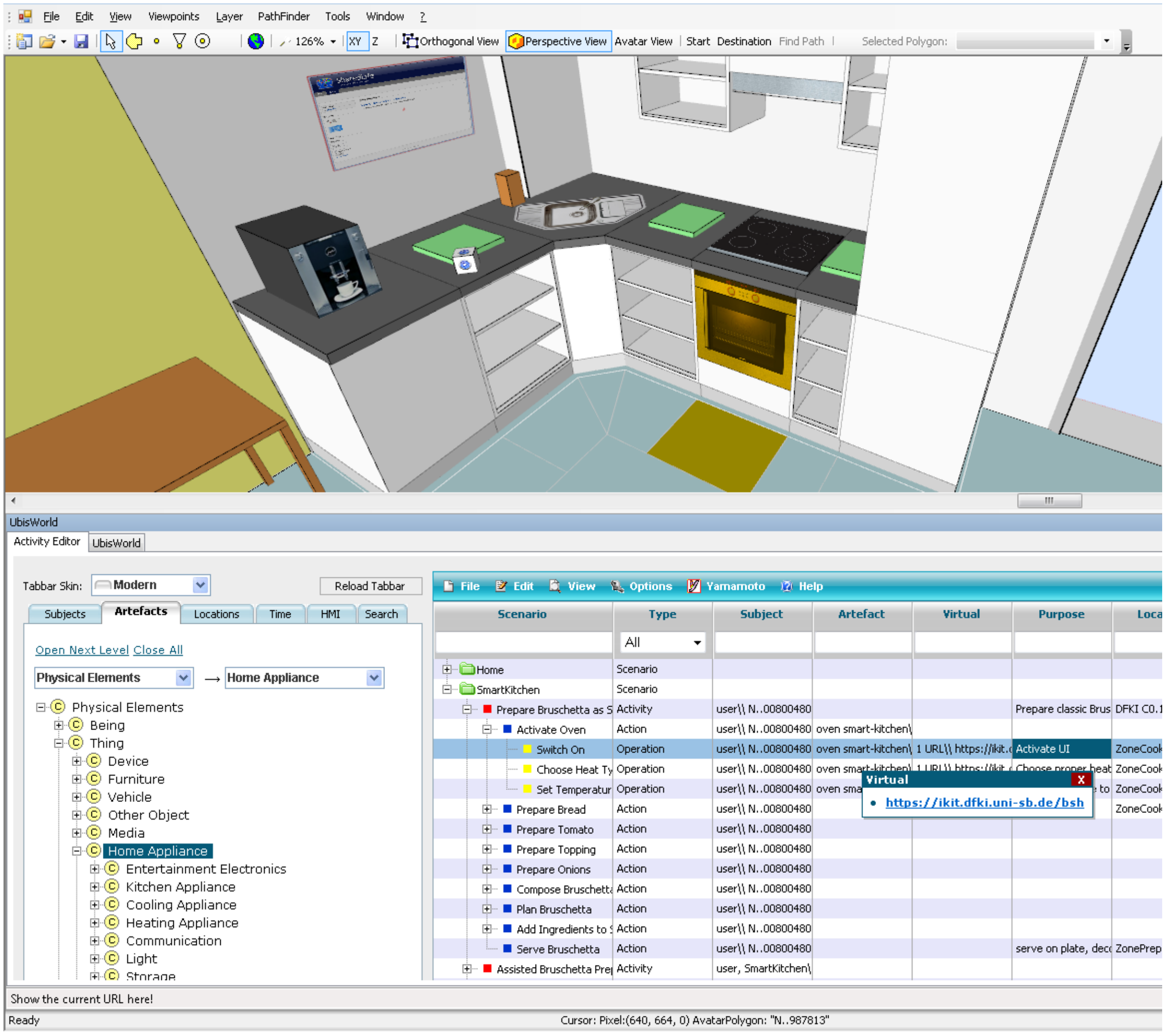
UbisLabel	Occurences
Display	[1] Device
TFT Display	[1] Vision
Smart Kitchen Display	[1] DELL 3008WFP
LCD Numerical Display	[1] Vision
LCD Graphical Display	[1] Vision
LCD Character Display	[1] Vision
Head-Moundted Display	[1] Wearable Computing

Figure 97: Searching elements in the UbisWorld ontology.

7.3 Activity Modeling

The *YAMAMOTO* toolkit has been developed to support the modeling and designing of instrumented environments according to the *GOAL* method, which has been presented in Chapter 6 of this thesis. Thus it provides a built-in **activity editor** that allows the modeling of activities according to the proposed process that is described in Section 6.3.3. We have suggested a formal representation for an activity model $\mathcal{M}$ in Section 6.3.3.6. The implementation of the activity editor's data model follows this suggestion, except for the needs, goals, and preconditions, which are represented as descriptive text instead of being part of the actional ontology.

The activity editor has been implemented as a Web application using HTML and Javascript, so that it can either be used stand-alone in any Web browser, or as part of the *YAMAMOTO* user interface, as shown in Figure 98. The editor's GUI is split in to frames; on the left-hand side, five tabs provide the necessary sets of concepts for subjects, artifacts, locations, times, and human-environment interaction instrumentation. The concepts are hierarchically organized as foldable trees, which are provided by the *UbisWorld* Web server and implemented using Javascript and AJAX. On the right-hand side, the activity model can be edited. The presentation of the activity model follows the tabular layout that has been proposed in Table 5. For the user interface of the activity editor we use the *TreeGrid* Javascript component that is provided by the *DHTMLX* company[55]. Internally, the Web server stores the activity model in a SQL database.



File Edit View Viewpoints Layer PathFinder Tools Window ?
126% XY Z Orthogonal View Perspective View Avatar View Start Destination Find Path Selected Polygon:
UbisWorld
Activity Editor UbisWorld
Tabbar Skin: Modern Reload Tabbar
Subjects Artefacts Locations Time HMI Search
Open Next Level Close All
Physical Elements → Home Appliance
File Edit View Options Yamamoto Help
Virtual
https://ikit.dfki.uni-sb.de/bsh
Show the current URL here!
Ready
Cursor: Pixel:(640, 664, 0) AvatarPolygon: "N..987813"

- Physical Elements
  - Being
  - Thing
    - Device
    - Furniture
    - Vehicle
    - Other Object
    - Media
    - Home Appliance
      - Entertainment Electronics
      - Kitchen Appliance
      - Cooling Appliance
      - Heating Appliance
      - Communication
      - Light
      - Storage

Scenario	Type	Subject	Artefact	Virtual	Purpose	Loca
	All					
Home	Scenario					
SmartKitchen	Scenario					
Prepare Bruschetta as S	Activity	user\\ N..00800480			Prepare classic Brus	DFKI C0.1
Activate Oven	Action	user\\ N..00800480	oven smart-kitchen\			
Switch On	Operation	user\\ N..00800480	oven smart-kitchen\	1 URL\\ https://ikit.c	Activate UI	ZoneCook
Choose Heat Ty	Operation	user\\ N..00800480	oven smart-kitchen\	1 URL\\ https://ikit.c	Choose proper heat	ZoneCook
Set Temperatur	Operation	user\\ N..00800480	oven sma	[illegible]	[illegible] to	ZoneCook
Prepare Bread	Action	user\\ N..00800480				ZoneCook
Prepare Tomato	Action	user\\ N..00800480				
Prepare Topping	Action	user\\ N..00800480				
Prepare Onions	Action	user\\ N..00800480				
Compose Bruschetta	Action	user\\ N..00800480				
Plan Bruschetta	Action	user\\ N..00800480				
Add Ingredients to S	Action	user\\ N..00800480				
Serve Bruschetta	Action	user\\ N..00800480			serve on plate, decc	ZonePrep
Assisted Bruschetta Pre	Activity	user, SmartKitchen\				

Figure 98: The activity editor is fully integrated in the Yamamoto user interface; artifacts like the oven can be highlighted in the 3-D view.

7.3.1 Modeling Activities, Actions, and Operations

The activity model is organized in scenarios, which can be used to separate activities from different projects or domains. The screenshot in Figure 98 shows the two scenarios "Home"

[55] http://dhtmlx.com/ visited 23.07.2009

and “SmartKitchen”. The latter is expanded, so that two activities “Prepare Bruschetta as Starter” and “Assisted Bruschetta Preparation” and their respective actions are visible.

The editor is operated via a context menu, as shown in Figure 99, which allows the user to add new scenarios, activities, actions, and operations. The enabled options depend on the selected row in the table. For example, if an activity is selected, actions can be added, but not operations. The table also preserves the order of the elements, which can be changed by moving rows up or down in the table.

The activity model is integrated with the 3-D view of the geometric model; all artifacts, locations, and instrumentation devices of a selected row (activity, action, or operation) that have been modeled in Yamamoto can be highlighted in orange color. In the example shown in Figure 98 the operation “Switch on”, which is part of the action “Activate oven”, refers to the oven, which is highlighted in orange color. Likewise, the location in front is highlighted.

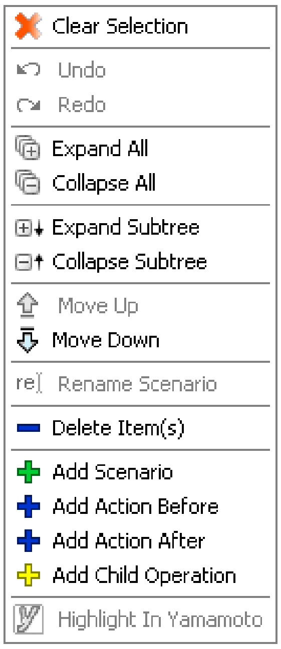


Figure 99: The context menu of the Activity Editor allows the user to create scenarios, activities, actions, and operations. Rows can also be moved up and down in the table.

7.3.2 Adding Subjects, Physical Artifacts, Locations, and Times

According to the definition given in Section 6.3.3.6, the activities, actions, and operations of the activity model $\mathcal{M}$ are represented as 6-tuples. These tuples are comprised of elements (subject, artifacts, object, locations, times, and interaction devices) that are taken from subsets of the ontology: physical, spatial, temporal, and actional.

The activity editor provides the elements for five of the six categories (except the objects) in separate tabs, namely “Subjects”, “Artifacts”, “Locations”, “Times”, and “HEI”. The “Artifacts” tab includes the whole physical branch of the ontology, whereas the “HEI” tab only includes the “Human-Environment Interaction” subtree. The “Locations” tab includes the geographic branch of the *UbisWorld*, as presented in Section 2.4.4.2.

Elements of the ontology can be added to the activity model in a drag-and-drop manner. By dragging a concept from one of the five trees to a row of the model, the element will be automatically inserted in the column that matches the source tab. E.g. if a *person* is dragged from the “Subjects” tab to an action in the activity model, this *person* will be added as subject for this action.

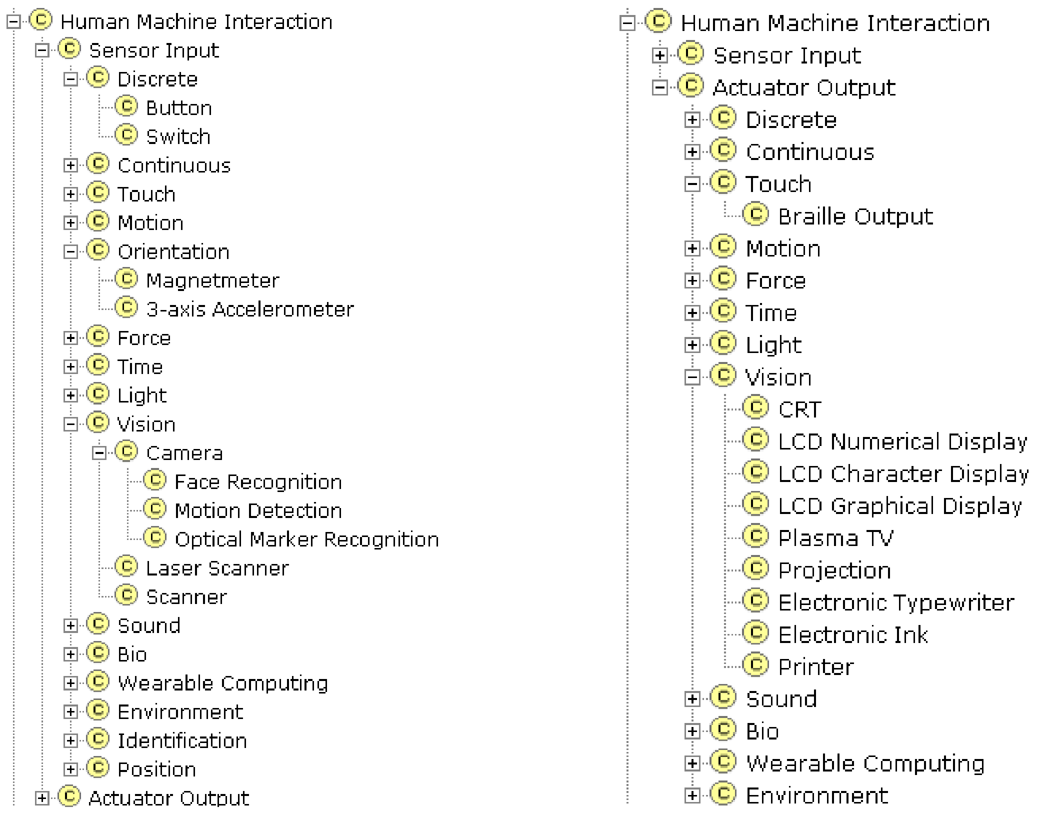
Concepts can only be added to the activity model’s fields, not removed. However, fields of the activity model can be double-clicked to edit their contents manually in order to delete elements. Multiple elements are listed in separate lines; the first line always gives a summary as a list of labels, whereas the lines below include the full UbisName of the elements.

7.3.3 Adding Virtual Artifacts

Virtual artifacts, such as digital user manuals, technical specifications, or Web interfaces, can be added to the selected row by entering their URL in the *“Enter virtual URL”*-field and pressing the *“Add”*-button. If the user moves the mouse over URLS in the activity model, a pop-up window opens and shows the URL as hyperlink.

7.3.4 Modeling Human-Environment Interaction

For the modeling of the environment’s instrumentation, we have extended the *UbisWorld* with concepts for input devices (sensors) and output devices (actuators), as shown in Figure 100 a) and b). The categories have been chosen with the intention that all known modalities and technologies can be added as subconcepts. For the future, we intend to fill each category with instances of existing hardware in order to support designers in their decisions. The categories for input and output are kept symmetrical, assuming that in most cases sensors exist to measure the effects of actuators (and vice versa).



a) Sensor/input concepts b) Actuator/output concepts

Figure 100: The ontology provides concepts and devices for Human-Environment Interaction.

7.4 Simulation, Evaluation, and Monitoring in Dual Reality

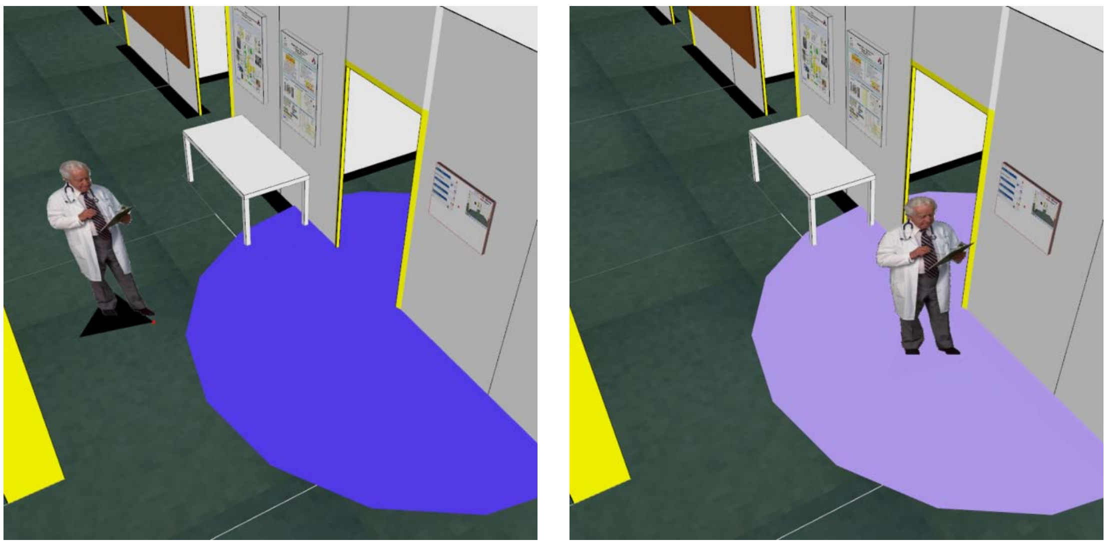
In this section, we present four tools that we have implemented in *YAMAMOTO* in order to simulate instrumented environments in virtual reality prior to their actual development and deployment. The benefits of such a simulation have already been extensively discussed in Section 6.3.5, and actual use cases will be presented later in Sections 8.2.6 and 8.5.3. Thus we focus here on the technical background and how to use the simulation tools.

7.4.1 Sending Virtual Bluetooth Proximity Sensor Events

In the scope of the development of the indoor navigation system *VISTO*, we required a means to identify users in front of public displays in order to provide a personalized user interface for them. We have solved the problem through Bluetooth dongles that continuously scan for Bluetooth devices. The scan returns the mobile devices' BT addresses, which are comparable to a *MAC* addresses in TCP/IP networks. The hardware dongles are operated by a small piece of software, which sends the recognized addresses as events to our instrumented environment's Event Heap proxy server via HTTP requests.

In order to simulate the Bluetooth scanner software, we have implemented virtual proximity sensors in Yamamoto that check if the avatar is located within their range. If the avatar is standing outside, they are quiet, see Figure 101 a). If the avatar moves inside the range, as shown in Figure 101 b), the virtual sensors become active (visualized in light blue) and begin to send events to the *SUPIE* environment. The events are equal to the events that are sent from the real sensors, so that for example the *VISTO* application responds to the avatar exactly as if a real user were present.

The sensors are modeled as sphere items (7.1.4.1) with their type set to Bluetooth. The sphere's radius determines their range. The sending of events can be activated by choosing "*Enable Sensor Events"* in the *"Tools"* menu. The host and port of the Event Heap proxy must be configured in the Yamamoto.ini file accordingly.



a) The avatar stands out of range. b) The Bluetooth sensor is active.

Figure 101: Virtual proximity sensors react to the avatar and send events to the *SUPIE* environment.

7.4.2 GPS-Receiver Emulation

In order to test and debug GPS-based navigation applications in the lab, *YAMAMOTO* can be used to emulate a real GPS-receiver unit. The term "emulator" is used for systems (implemented in hardware or software) that duplicate the functions of original hardware in order to behave like (and in fact appear to be) the original system. In our case, *YAMAMOTO* includes a software module that imitates the output stream of a GPS-receiver unit. Any GPS-based mobile application that is connected to this output stream will believe that it receives real GPS position data. The advantage is that we can set the position to where ever we want to pretend to be in order to test the mobile application's behavior.

The GPS emulation can be enabled in *YAMAMOTO* in the *"Tools"* menu by choosing *"Enable GPS-Emulation"*. By default, serial port *COM1* is opened with a bitrate of 4800 baud. Other settings can be configured in the "GPS" section of the *yamamoto.ini* file, see Table 7 for a listing. The user can then control the simulation by moving the avatar through a geo-referenced (7.6.1) model. *YAMAMOTO* converts the avatar's model coordinates into geographic coordinates (7.6.2). These coordinates are encoded into *NMEA 0183* sentences. The *NMEA 0183*[56] Interface Standard has been developed by the *National Marine Electronics Association* and defines electrical signal requirements, data transmission protocol and time, and specific sentence formats for a 4800-baud serial data bus. It is the industry standard for communication between GPS-receiver units and navigation applications.

Yamamoto creates a typical set of three *NMEA* sentences that represent a GPS position fix (*GPRMC*), GPS satellite visibility (*GPGSV*), and compass direction (*HCHDG*). If the simulation is active, these three sentences are sent once per second to the serial port. Please note that the *GPGSV* sentence is always the same and does not depend on the avatar. It is just included to pretend satellite reception for the mobile application. Example sentences are shown in Table 8.

Table 7: Serial port configuration parameters.

Key	Value	Default
COMPort	COMn	COM1
Baudrate	Integer-value	4800
Encoding	ASCII / UTF-8	ASCII
Parity	None / Even / Odd	None
StopBits	Integer-value	1
DataBits	Integer-value	8
Handshake	None / RequestToSend / RequestToSendXOnXOff / XOnXOff	None

Table 8: Example NMEA 0183 sentences that are emulated.

```
$GPRMC,191410,A,4735.5634,N,00739.3538,E,0.0,0.0,131106,0.4,E,A*19
$GPGSV,1,1,04,01,40,083,45,02,17,308,45,03,07,344,45,04,22,228,45*99
$HCHDG,010,,0.2,E*99
```

[56] http://www.nmea.org/content/nmea_standards/nmea_083_v_400.asp, visited 20.07.2009

7.4.3 VNC Display Connection

As stated earlier in Section 0, *YAMAMOTO* display objects partially implement a VNC (Virtual Network Computing) client. VNC has been originally developed at the ORL (Olivetti & Oracle Research Lab) to remotely control another computer (server) on a local machine (client) using mouse and keyboard. The content of the graphical display of the remote machine is grabbed and transferred via network to the local computer, based on the RFB protocol (Richardson, 2005). In the case of *YAMAMOTO*, the content is mapped as image texture on the virtual display. What is required are the installation of a VNC server on another computer (host) and the setting up of the name or IP-address of the host in the display object properties. The default port is set to 5900, but other ports can be also specified. It is recommended to protect the server by a password, which also has to be configured in the display properties. Once the server has been set up, the VNC connection can be activated. In order to save network bandwidth, some implementations support to scale the display content down to 50% or 25% of the original size. Currently no interaction with mouse or keyboard is implemented, but it would be theoretically possible to do so by clicking with the mouse on the virtual display in *YAMAMOTO*.

7.4.4 YCI Yamamoto Control Interface

The *Yamamoto Control Interface* (YCI) allows external applications to remote-control the *YAMAMOTO* modeling toolkit. The *YAMAMOTO* avatar can be manipulated from 3rd party software in order to visualize and track the position of a user in the real-world. In the other direction, the position of the *YAMAMOTO* avatar can be read to simulate a positioning system. Furthermore, other *YAMAMOTO* objects, such as shelves, tables, or virtual displays, can also be manipulated. The YCI also provides commands to convert coordinates between model units (pixels) and geographic units such as meters, Longitude, Latitude, and altitude.

The YCI has to be manually enabled by the user by selecting “Enable TCP Socket” from the “Tools” menu after the model has been loaded. Now an external application can connect to a running instance of *YAMAMOTO* via a TCP/IP socket connection either on the same machine (IP-address 127.0.0.1 or “localhost”) or remotely from the network. The default port is *4711*, other ports can be specified in the *YAMAMOTO* configuration file (yamamoto.ini). There can be up to 10 clients connected to Yamamoto at the same time.

The command keywords are not case sensitive, but object names are case sensitive. For example, both “SELECT Box1” and “select Box1” will work, but not “select box1”. Each command sent to *YAMAMOTO* should end with a *newline* (LF) character ‘\n’. Additional *carriage return* (CR) characters ‘\r’ are ignored. All return values that are sent back from *YAMAMOTO* also end with a *newline* character. We use the *US* decimal separator which is a point, e.g. PI is *3.14159.*

To access any object, it has to be first selected by issuing a “*SELECT* <name>” command. The name refers to the name attribute of any object, preferably a unique *UbisWorld*[57] identifier, and is case sensitive. The special name “*Avatar*” refers to the avatar. The following *GET* and *SET* commands will refer to the previously selected object only. The return values are “OK” or “*ERROR*”, if no object with the name could be found in the model. To manipulate the

[57] UbisWorld website: http://www.ubisworld.org

selected object, send a “*SET <attribute> <value>\n*” string. Finally all attributes should be accessible as defined in the YML Document Type Definition (DTD). In the current release only a small subset of attributes is supported. For convenience, attributes and values can be concatenated, for example:

“SET x 100 y 200 z 300” sets the x attribute to 100, y to 200, and z to 300. Return values are ”*OK*” or “*ERROR*” if any unknown attributes are specified in a set command.

To read out the current attribute values of the selected object, send a “*GET <attribute>\n*” string to the socket connection. For convenience, attributes can be concatenated:

“GET x y z” results in return value *“100 200 300”*. Unknown attribute names result in a “*null*” value at the corresponding position. For example *“GET x test z”* would result in *“100 null 300”*.

To convert length units between model units (pixel) and meters one can use the “*ModelToMeter <model>\n*” and “*MeterToModel <meter>\n*” commands, which return meters and model units, respectively. If no scale is set in the current model, an error message is returned. To convert locations between model (pixel) units and geographic units (degrees) use “*ModelToGeo <x> <y> <z>\n*” and “*GeoToModel <longitude> <latitude> <meter above sea level>\n*”. If no geographic reference is set in the current model, an error message is returned.

Here we list the attributes of the Avatar and Box objects:

Avatar Object

- Selection: “SELECT Avatar*n*”
- Attributes:
    - x (model units)
    - y (model units)
    - z (model units)
    - rotation (degrees 0-360)
    - image (URL string)

Box Object

- Selection: “SELECT <*box name*>*n*”
- Attributes:
    - x (model units)
    - y (model units)
    - z (model units)
    - width (model units)
    - depth (model units)
    - height (model units)
    - orientation (degrees 0-360)

7.5 Hierarchical Modeling at Multiple Scales

Our physical environment is constantly evolving, at a slow pace though. Therefore it is not only for memory considerations, but also for maintenance efforts, to adopt a 'divide-and-conquer' strategy for the creation and operation of a distributed, large-scale environment model for pedestrian navigation.

We divide the world into a hierarchical tree structure, where each node refines one object of its parent node on a higher level of detail, or finer granularity in terms of the UbisWorld ontology. Figure 102 shows the campus model of Saarland University in Saarbrücken, Germany. On top of the hierarchy, the root model represents the main roads, footpaths, and the shapes of buildings by their outline. If a building has multiple entrances, the corresponding floors are modeled as separate levels. On the next level of granularity, each building is represented by its own refinement model. These models typically include rooms, staircases and elevators. The leaf nodes of the tree finally represent single rooms on a high level of detail, for example the kitchen model includes a detailed representation of each shelf.

Each refinement-node may specify its geometry using an own coordinate system, as described in Jiang and Steenkiste (2002).

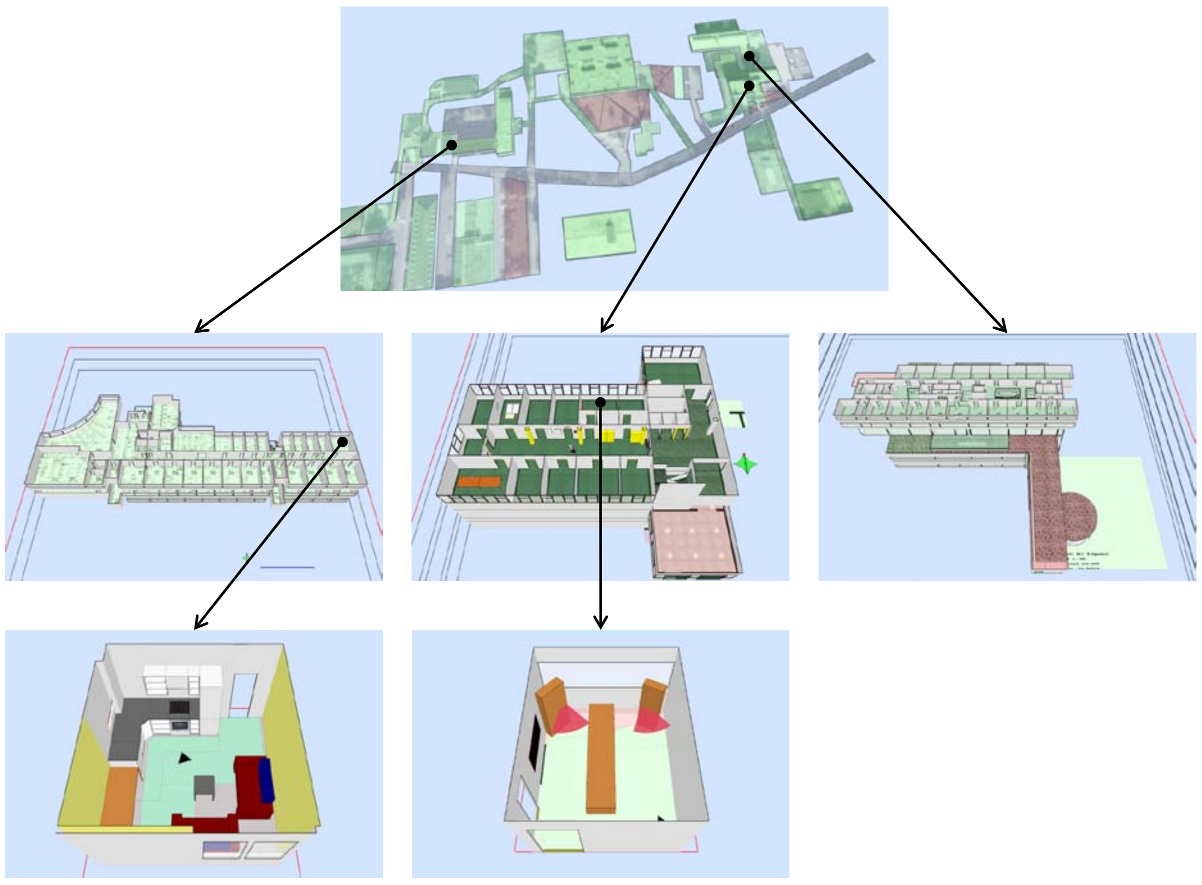


Figure 102: The model tree for the Saarland University campus shows the outdoor survey model, three refinement models for the computer science buildings, and two detailed room models.

The hierarchical approach has several benefits for rendering, maintenance, and route finding. The modeling on different levels of detail effectively reduces the number of polygons in the rendering pipeline; it also saves memory and processing resources. Furthermore, the model tree can be easily distributed over multiple web servers, so that building and room models can be individually provided and maintained by their inhabitants. Last but not least, the route finding service *PathFinder* also benefits from this strategy, since it effectively reduces the search space for the A* algorithm; models are dynamically loaded at runtime, if needed.

7.5.1 Relations between Models

In the YML data model, we use bi-directional links between parent and refinement models. Each polygon (<Object> element) can have an optional <SubModel> element that specifies a refinement model in its "URL" attribute. Vice versa, the <Model> element of each model has a pointer towards its parent-model URL ("rootURL" attribute) and the polygon that it refines ("rootObject" attribute). Furthermore, the <Model> element describes how the refinement model is mapped into its parent model in terms of translation ("xShiftAsSubModel", "yShiftAsSubModel" attributes), rotation, and scale ("rotationAsSubModel", "scale-AsSubModel" attributes). We will show in the next section how this mapping can be done.

By labeling edges with unique names, multiple representations of the same edge can be defined in different nodes of the model tree. These "cross-model" gates allow the *PathFinder* algorithm for route planning beyond the boundaries of the current granularity level. For example, a route can be found from an office in the computer science building "E1 1" to a conference room at the DFKI building "D3 1" via the main road. Using names instead of coordinates offers the freedom to abstract details from subsumed polygons on a higher level of granularity; the entrance on the outline of a building in the outdoor plan does not have to exactly match the entrance's coordinates in the detailed building model. The label of a cross-model gate is defined in YML by the "label" attribute of the <Connection> edge.

7.5.2 Graphically Assigning a Submodel

Using *YAMAMOTO*, the refinement of some polygonal object, such as the outline of a building through a more detailed model of this area, e.g. an interior model, can be a done in three simple steps:

1. The polygon that represents the outline of the building has to be selected. If no submodel has been assigned yet, this can be done by choosing "*Assign existing submodel*" from the "*Edit*" menu to select an existing .yml model. The editor now tries to automatically scale and rotate the refinement model according to its geographic reference points or alternatively its scale and orientation information. Otherwise choosing "*Edit submodel*" allows to re-align the current submodel. The submodel is displayed as overlay within the parent model.
2. The building-model overlay can now be aligned to match the outline, as shown in Figure 103. The submodel can be moved by left-clicking on the model and holding the mouse button. The yellow markers in the corners of the model allow scaling it. The model can be rotated by pressing and holding the "Shift" key. Now the markers change their color to red, and allow rotating the model.

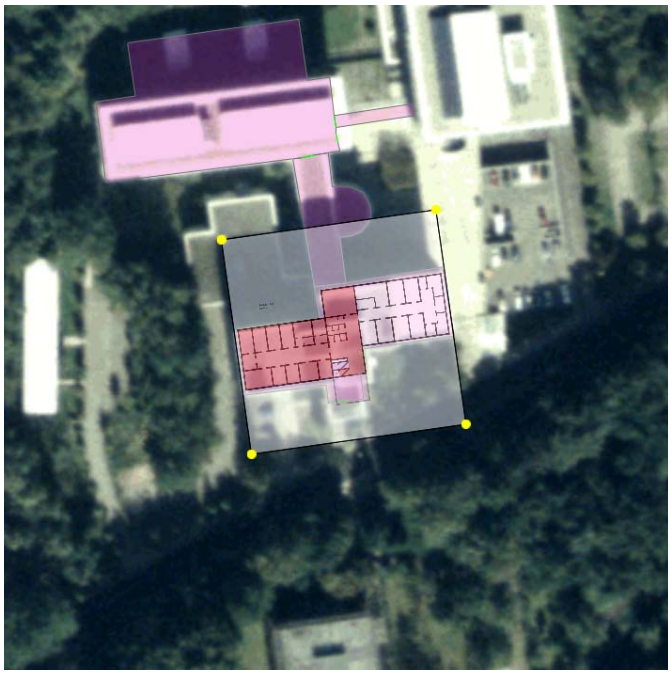
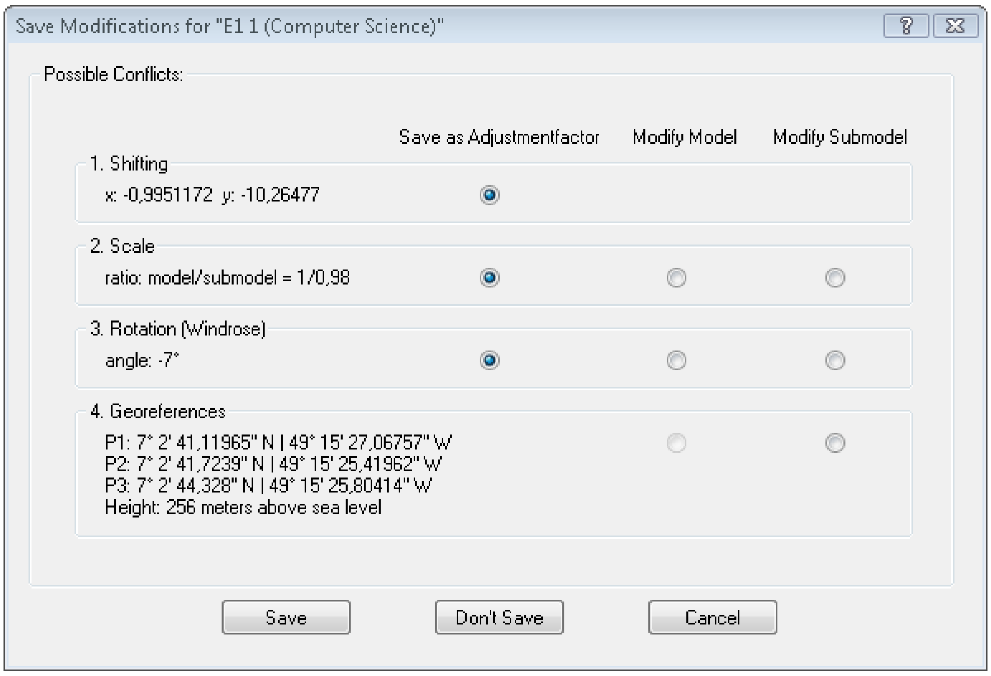


Figure 103: Refining a building object by its interior model.

3. Selecting the pointer icon from the toolbar fixates the submodel in its current position. Now a dialogue window opens and displays the changes that have been made, as shown in Figure 104. The resulting changes can be stored as correction factors in the header of the submodel, so that its original settings for scale, wind rose, and reference points remain untouched. If desired, the changes for scale and rotation can also be directly applied either to the parent model or the submodel. If the orientation of a building model is not indicated in the floor plan, the recommended option is to modifying the submodel.


Save Modifications for "E1 1 (Computer Science)"

Possible Conflicts:

	Save as Adjustmentfactor	Modify Model	Modify Submodel
1. Shifting x: -0,9951172 y: -10,26477	●		
2. Scale ratio: model/submodel = 1/0,98	●	○	○
3. Rotation (Windrose) angle: -7°	●	○	○
4. Georeferences P1: 7° 2' 41,11965" N | 49° 15' 27,06757" W P2: 7° 2' 41,7239" N | 49° 15' 25,41962" W P3: 7° 2' 44,328" N | 49° 15' 25,80414" W Height: 256 meters above sea level		○	○

Figure 104: The manual alignment of the submodel results in adjustment factors.

7.5.3 The Model Tree Window

The model tree window is situated at the lower right part of the *YAMAMOTO* user interface and helps the user get an overview of the hierarchical structure, as depicted in Figure 105. It clearly shows the actual model hierarchy, beginning from the root node. By clicking on a node with the right mouse button, a context menu appears that allows to open (and close) nodes in the editor. It also shows some useful statistical information about each node, such as its URL, number of polygons, and number of layers.

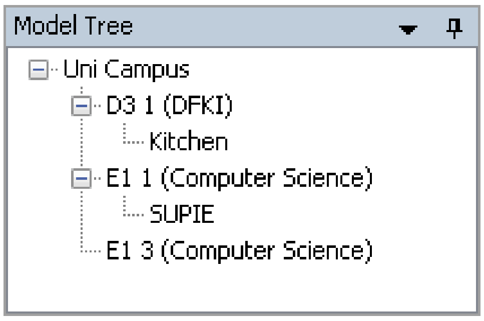


Figure 105: The model tree window gives an overview about the model hierarchy.

7.6 Geo-referencing the Model with Longitude and Latitude

One special requirement for pedestrian navigation systems is the ability to seamlessly move between indoor and outdoor environments. However, indoor and outdoor models are very different. Architects describe the layout of building structures by measuring the length of each element and usually orient their design on a rectangular grid. In mathematical terms the Euclidean space is the ideal representation for the geometry of a building. Cartographers consider the fact that our world is a sphere, so they use a spherical coordinate system where each point is defined by its radial distance from a fixed origin; the zenith angle from the positive z-axis, and the azimuth angle from the positive x-axis. In geography, most common geodetic reference systems, such as *WGS84* that is used for the global satellite positioning system *GPS*, approximate the earth as an ellipsoid and define locations by a triple of their altitude, Latitude (as the complement of the zenith angle) and Longitude (as the complement of the azimuth angle); see Appendix D for more information.

Considering only small local areas, such as a university's campus, the deviation between the Euclidean and spherical coordinate systems is small so that it can be neglected for navigation purposes. The challenging issue for a combined indoor-outdoor model is how to translate position information between both coordinate systems. In a typical case, a traveller is using outdoors a combination of a GPS satellite receiver and geographical map data to navigate to the destination building. Upon entering the building however, the satellite connection will break down and the traveller has to rely on a specialized indoor positioning system, based on navigational beacons, and floor map data. We have already introduced the *LORIOT* positioning system that combines infrared with RFID beacons as an example in Section 7.1.4.1. The crucial point for a seamless transition between both maps and positioning techniques is the ability to convert geographic coordinates of Longitude and Latitude in Cartesian map coordinates, and vice versa.

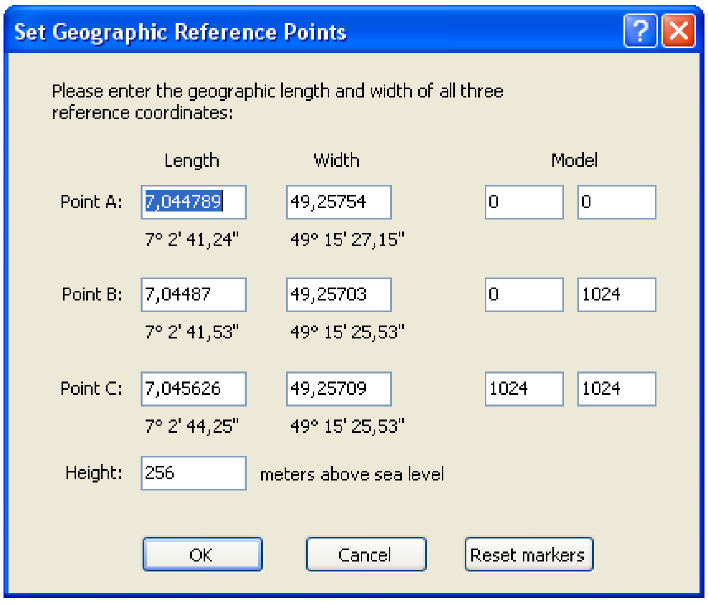
There are several ways to define the relationship between indoor and outdoor models so that a definite mapping is possible. We can either specify three reference points in both models, or alternatively provide one base point (translation), together with scale and rotation information. For convenience, the Yamamoto editor provides tools for each of the four measures, even though they are ambiguous.

For outdoor maps, geographic references can be easily derived for certain points from a land registry office. In Germany, the Gauss-Krüger coordinate system is used. It has to be converted into the WGS84 or ETRS84 system (Appendix D), if compatibility with the GPS system is desired. Moreover, a free coordinate conversion tool called *GEOTRANS* is available from the National Geospatial-Intelligence Agency[58], U.S.A. (NGA). In case that no geo-referenced maps are available, even aerial photographs from *Google Earth*[59] can be good enough if the image resolution is about 1 meter.

7.6.1 Setting Geographic Reference Points

The user can manually set three reference points in the model that define both model units and geographic coordinates in Longitude and Latitude (see Figure 106). These points span one basis for a vector space in model units, and another basis for a vector space in geographic units. Of course, the mapping of spherical coordinates into Euclidean space is only valid for small local areas.

Choose "*Set Geo-References*" from the "*Edit*" menu to open the geo-reference dialog window as shown in figure F. You can set the Longitude and Latitude and according model coordinates for three reference points (A, B, C). The coordinates are entered in decimal-degree notation and are also shown in sexagesimal (minute/second) notation. It is also possible to specify the height of the model's origin in meters above sea level. By default, the reference points will be set to the corners of the model's space, which is usually defined by the size of the backdrop image texture.


Set Geographic Reference Points

Please enter the geographic length and width of all three reference coordinates:

	Length	Width	Model	
Point A:	7,044789	49,25754	0	0
	7° 2' 41,24"	49° 15' 27,15"		
Point B:	7,04487	49,25703	0	1024
	7° 2' 41,53"	49° 15' 25,53"		
Point C:	7,045626	49,25709	1024	1024
	7° 2' 44,25"	49° 15' 25,53"		

Height: 256 meters above sea level

Figure 106: Setting reference points for conversion between geographic and model coordinates.

[58] NGA: http://earth-info.nga.mil/GandG/geotrans/
[59] Google Earth

The same procedure applies for indoor models. However, typically no reference points are known inside of the building. In that case the Yamamoto toolkit provides an alternative solution based on the hierarchic modeling approach, as described earlier in Section 0. An indoor model of a building can be visually aligned through translation, scale, and rotation operations to match its outline as shown on an outdoor image. Now the indoor model can automatically derive its geographic reference points from the outdoor model, provided that the latter has already been referenced. Of course the accuracy will be fairly limited, depending on the resolution of the outdoor model's image texture. Yet it allows for a seamless transition between indoor and outdoor spaces based on geographic coordinates.

7.6.2 Coordinate Conversion

If a model has valid geo-references, coordinates can be converted from model units based on the texture into geographical coordinates. They can be accessed through Yamamoto in several ways. Firstly, the current position of the mouse pointer will be shown in the status bar in both model and Longitude/Latitude units, so that geo-coordinates can be easily read off for any point in the model. Secondly, geo-information is shown in the Point properties and Item properties. The number format can be switched between sexagesimal (minute/second) and decimal notation via the "Longitude/Latitude Format" settings in the "View" menu.

The *.yml* file also contains a 4x4 matrix and its inverse to conveniently map in external applications (such as 8.4) model coordinates into geographic coordinates and vice versa. If a model coordinate (x, y, z) has to be converted from model units into geographic Longitude, Latitude and altitude, a vector *p* = (x, y, z, 1.0) can be multiplied with the *ModelToGeo* matrix *M* as shown in Formula (1):

$$(1)\;\; M \cdot p = g \qquad \text{or:} \quad \begin{pmatrix} m00 & m01 & m02 & m03 \\ m10 & m11 & m12 & m13 \\ m20 & m21 & m22 & m23 \\ m30 & m31 & m32 & m33 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} long \\ lat \\ alt \\ 1 \end{pmatrix}$$

The resulting vector *g* = (Longitude, Latitude, altitude, 1.0) contains the geographic coordinates. Vice versa, geographic coordinates can be converted into model coordinates by multiplication with the *GeoToModel* matrix *M'* (note that *M'* is simply the inverse matrix of *M*: $M' = M^{-1}$) as shown in Formula (2):

$$(2)\;\; M' \cdot g = p \qquad \text{or:} \quad \begin{pmatrix} m'00 & m'01 & m'02 & m'03 \\ m'10 & m'11 & m'12 & m'13 \\ m'20 & m'21 & m'22 & m'23 \\ m'30 & m'31 & m'32 & m'33 \end{pmatrix} \bullet \begin{pmatrix} long \\ lat \\ alt \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Optionally, the *.yml* file can also be created with additional geo-coordinates for each <item> element by choosing "*Save with geographic information*" from the "*File*" menu.

7.6.3 Aligning the Model to North using the Wind Rose

Whereas on the one hand, outdoor maps are usually aligned to the geographic north direction, on the other hand, indoor maps, such as floor plans of buildings, are typically aligned to the building structure. Therefore it is often necessary to indicate the orientation of the model. Traditionally, maps show a wind rose to indicate the cardinal directions. Similarly, the user can choose “Add Wind Rose” from the “Edit” menu to add a wind rose object to the model. The wind rose can be dragged with the mouse to any position in the model and the red dot can be moved to set the north direction (see Figure 107). It is possible to model contradicting information about the north direction if the orientation of the compass does not match the geographic reference points. It is recommended to use the compass only for indoor models in order to derive geographic reference points from an outdoor model through the hierarchical modeling process described above.

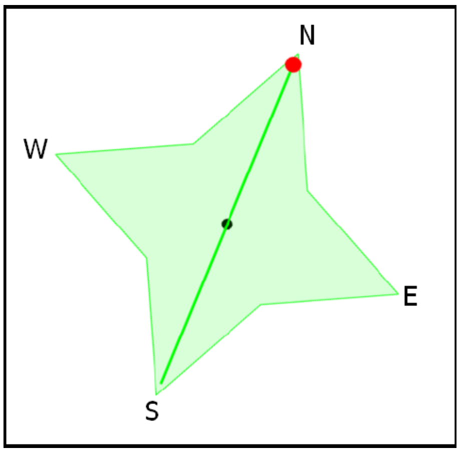


Figure 107: The wind rose object can be used to indicate the orientation of a model

7.7 Wayfinding

YAMAMOTO has been developed with a strong focus on wayfinding assistance, which naturally requires a route finding component. For this purpose, we have implemented the PathFinder module. It has been designed to return a path between any two points in a mesh of adjacent polygons, such as a *YAMAMOTO* model. The route is only allowed to cross edges if they are annotated to be passable and should directly follow the line-of-sight unless turning points are caused by obstacles like walls. Our algorithm aims towards paths that appear naturally for pedestrians, which means to use shortcuts if possible while keeping a safe distance from walls and corners. The typical behavior of pedestrians to radically shorten their path often manifests itself in visible trails on greens. Figure 108 shows a heavily worn-out carpet in a museum setting that gives evidence of this behavior. It gives a somewhat “statistically significant” proof that people pass the corners in a distance of about one foot length.

In the remainder of this section, we will introduce how to find and visualize routes in *YAMAMOTO* and finally explain the details of the algorithm. We recommend reading about the algorithm because it will help develop a better understanding of the relationship between the models, in particular the segmentation of complex spaces, and the resulting paths.

Figure 108: Worn-out carpet in the archeological museum of Istanbul.

7.7.1 Route Finding in Yamamoto

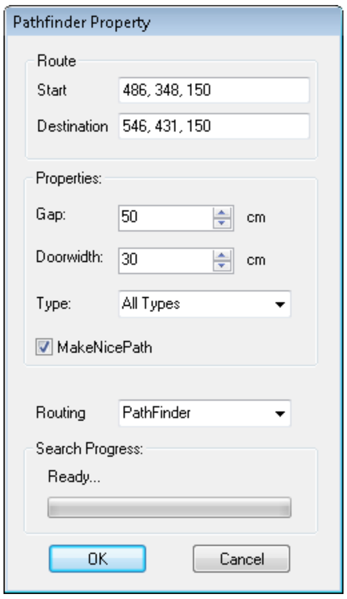
The *Pathfinder* component provides a route finding algorithm for *YAMAMOTO* models and can be accessed through the Pathfinder dialog box shown in Figure 109. If a route exists, it will be highlighted in green color and arrows appear at turning points, as shown in Figure 110. In order to find and visualize the shortest path between two points in the current model, proceed in three steps:

1. Select the layer of the starting point in the layer controller, and then choose "Set Start" from the "*PathFinder*" menu or press "*Start*" in the toolbar. Now click into the model to set the starting point for the route.
2. Select the layer of the target point in the layer controller, and then choose "Set Destination" from the "*PathFinder*" menu or press "*Destination*" in the toolbar. Now click into the model to set the ending point for the route.
3. Select "*Find Path*" from the "*PathFinder*" menu or the toolbar. The Pathfinder dialog box opens. Now click "*OK*" to start the routing algorithm with the default settings. If a route exists, it will appear as a green line in the model. Press "*Cancel*" to close the dialog box.

The Pathfinder dialog box provides some additional parameters to modify the search:

- Gap. This distance defines the minimum gap between the path and walls. If the gap value is set to high, no route can be found through narrow passages or doors.
- Type. Choosing "*All Types*" ignores the accessibility attribute of polygons. If one of "*Pedestrian*", "*Car*", or "*Wheelchair*" is selected, the path will cross only polygons with a matching accessibility. All other polygons will be ignored for the routing algorithm.

- Select "*Nice Path*" option for an optimized path.
- Routing. Chooses between the "*PathFinder*" and "*PathFinder2*" search engines. "*PathFinder*" is faster, but limited to the current model, whereas "*PathFinder2*" is able to find routes across multiple models, but slower, since it has to load and parse each model separately. It also allows for searching locations by name in the "*Start*" and "*Destination*" fields.


Pathfinder Property

Route

Start	486, 348, 150
Destination	546, 431, 150

Properties:

Gap:	50	cm
Doorwidth:	30	cm
Type:	All Types	

[✓] MakeNicePath

Routing: PathFinder

Search Progress:

Ready...

Figure 109: The Pathfinder dialog defines start, destination, and search parameters.

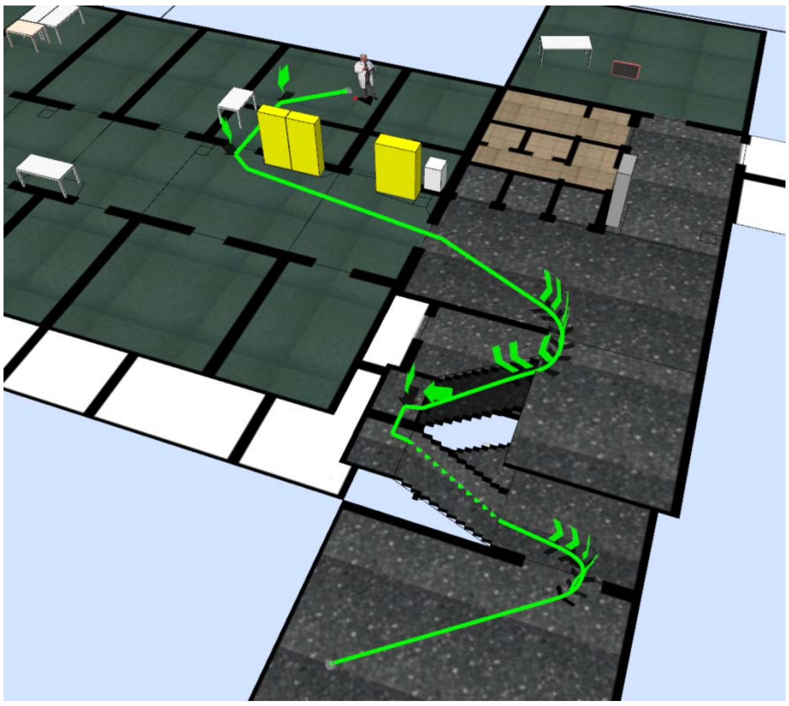


Figure 110: Example of a route that includes stairs.

7.7.2 The PathFinder Algorithm

As already introduced above, the route finding algorithm has been designed to return a shortest path between any two points in a model. Our approach is to reduce the search in the 2.5-dimensional space to a simpler graph problem. Therefore we first pre-compute a path network from the polygon-based model. Similarly to the hierarchical model tree presented in Section 7.5, multiple models result in multiple path networks that are hierarchically linked together. The 'divide-and-conquer' strategy of multiple levels of detail allows for an efficient searching strategy where detailed building and room geometry is dynamically loaded only if needed. We will now give a short overview of the algorithm before we go into the details of each stage.

7.7.2.1 *Overview of the Algorithm*

The path network graphs $G_i(V, E)$ are constructed from the polygon models as sets of relevant turning points *V* and line segments *E* between them. We make the assumption that the relevant points are most likely determined by those corners where two walls include an angle greater than 180 degrees. Thus in the first step a set of relevant points *V* is automatically created from these corners. Segments that cross polygon edges are only valid if they are annotated to be passable and should directly follow the line-of-sight unless turning points are caused by obstacles like walls. Hence an edge *e(p,q)* exists in *E* if (and only if) both *p* and *q* lie inside adjacent polygons and the edge *e* intersects only 'passable' polygon borders. We also include the endpoints of passable edges (such as doors) in *V*. The points are moved slightly away from the walls inside the room in order to keep a natural distance for the pedestrian. Now an *A** search (Hart et al, 1968) for the shortest path in *G(V, E)* is conducted. We can apply additional weights to the edges during the search to consider individual preferences of the user, e.g. to prefer a lift over stairs. For reasons of computational complexity, the construction of *P* does not necessarily include any point that is required for the shortest path. So the result of the search in *G* might not be the 'true' shortest path in the polygon model, and we have to optimize it in a third step. Here we try to straighten the path to follow the line-of-sight wherever it is possible. Again, this optimization problem can not be solved in non-exponential time, so we use a simplified algorithm that works well for typical cases. During the modeling process, care has to be taken to avoid situations that cause the optimization to fail.

7.7.2.2 *Constructing the Path Network Graph*

The path network graph has to meet two quite opposing requirements: on the one hand, the graph has to be detailed enough to include all possible routes, including the shortest one. On the other hand, the complexity of the A* search heavily depends on its size, so it should be as small as possible. This contradiction asks for a reasonable tradeoff between quality and efficiency of the search (Waßmuth, 2006).

In order to ensure the quality as well as fast results of the search, we have to construct the path network in an intelligent way. Whereas a naive approach would define a uniform grid of points, our strategy is to set the points only where they are most likely to be needed, depending on the model.

The algorithm is based on the following graph data structure:

- The **path network graph *G(V, E)*** is defined by a set of vertices *V* (turning points) and edges *E* (line segments).

The possible paths in the graph *G* represent corresponding paths in the model, which consist essentially of straight line segments (edges *E*) and only change their direction at certain turning points (vertices *V*). The vertices are only created in positions where the pedestrian might have to change their direction, which is the case at concave corners of the model and between polygons near doors and passages.

7.7.2.3 *Turning Points*

We have already explained in the introduction of this section that a path for pedestrians has to keep a safety distance to corners and walls, whereas the geometrically shortest path cuts the corners like a rubber band. Hence the turning points are created not directly at the mentioned corners in the model, but moved inside the polygon with an offset of about two foot lengths. Figure 111 shows an example scenario, where two rooms are connected by doors to an L-shaped corridor that is segmented in two parts. Figure 112 depicts the resulting gap to the wall shaded in grey and the set of potential turning points *V*.

In non-convex polygons corners with an angle greater than 180 degrees are possible and pose an obstacle to the pedestrian, so that they require a turning point in order to bypass them. Also obstacles, caused by box objects in *YAMAMOTO* models, require turning points at their corners. Ideally, a smooth path would follow a segment of a circle to bypass corners, but we cannot consider this in our path network graph. We will describe a post-processing method to interpolate a path with Bezier curves in the *VISTO* application in Section 8.2.

Passable edges also require turning points and we discern between doors and passages based on the edge attributes in *YAMAMOTO*. Additionally, short edges are also automatically treated as doors. In the case of doors, we assume that the pedestrian should pass through the middle of the door. Hence we create a turning point in the center of the edge. All other cases are treated as passages, where adjacent polygons form a larger space. This often occurs if rooms or passageways are segmented into smaller polygons during the modeling process. Such passages should not influence the path, so we create two turning points at each end of the edge. The difference can be seen in Figure 112; whereas the edges that represent doors have points in their middle, the segmentation edge of the corridor does not.

Elevators have to be treated separately, since their vertical axis of motion has no geometrical representation in the *YAMAMOTO* model. Instead, the elevator booth is represented by one polygon per level. These polygons are attributed as “elevator” and grouped by a unique name. Elevators can easily be mapped into the graph by introducing additional turning points in the center of each elevator booth polygon and adding vertical edges between them. As a side effect, the connections between elevators are always shorter than staircases, so that they will be automatically preferred.

The links between models on different levels of detail in the model hierarchy have to be treated in a similar way. Edges that link two models are annotated as external links in the model and identified by a label. During the search, they are represented by a special type of turning point in their center and cause the A* search algorithm to dynamically load and include the connected model, if such a point is traversed.

Finally, the start and destination points of the current search request are added to *V*.

7.7.2.4 *Line Segments*

In order to minimize the branching in the graph, we try to sparingly use edges in the graph. Hence we only include edges between turning points of neighbor polygons; between two vertices V1 and V2 exists an edge e if and only if the corresponding polygons P1, P2 are either identical or neighbors (they share a passable edge), and the line segment between P1 and P2 is passable. A line segment is passable if it does not cross any non-passable edges in the original model. The algorithm performs an intersection test for each edge, where the 2.5-dimensional geometry is projected in two-dimensional space. Edges may be annotated as passable from left or right to represent situations like one-way streets or exit-only doors, so the direction is also considered. The resulting edges of our example scenario are shown in Figure 113.

This strategy improves the search speed, but the downside is that some direct paths are missing in the graph whereas a direct route exists in the original model. This may cause undesired turns and deviations, so that the result of the A* search has to be optimized in a post processing step.

7.7.2.5 *Adapted A* Search*

PathFinder uses an adapted A* search algorithm on the path network graph *G*. We use the Euclidean distance between the current node and the destination as heuristic function to travel the graph, even if this is not the best strategy in multi-level buildings. In a building, the staircases should be considered by the heuristic as well. If certain areas of the network should be preferred or avoided, edge weights can be applied. For example, edge weights $w > 1$ increase the total cost of a path, so that routes along these edges appear longer than they actually are and other routes are preferred as result, even if they are actually detours.

The adapted A* algorithm initially loads only those models that contain the start and ending points of the route, in order to keep the search space small. If the search expands a node that represents a link to another model, this model will be dynamically loaded and added to the path network graph *G*. The search will now continue within the expanded graph and consider routes through the submodel. For example, if the search algorithm reaches the entrance of a building in a street map, the detailed model of the building will be loaded in order to find a way through it.

7.7.2.6 *Optimizing the Resulting Path*

The resulting route of the A* search is often not the shortest path due to the construction of the path network graph G that only contains points at corners. Furthermore, the search algorithm only moves between points of neighboring polygons. In the case of a segmented corridor, these restrictions can cause a route that follows one side of the corridor, then crosses the corridor, and continues along the other side. This effect can be mitigated by removing points from the path to straighten it, if possible. By doing so, new edges are considered that have originally not been included in the path network graph G. Please see Figure 114 for an example scenario. The dotted line is the result of the A* and not the optimal solution, whereas the continuous line has been optimized by removing one point.

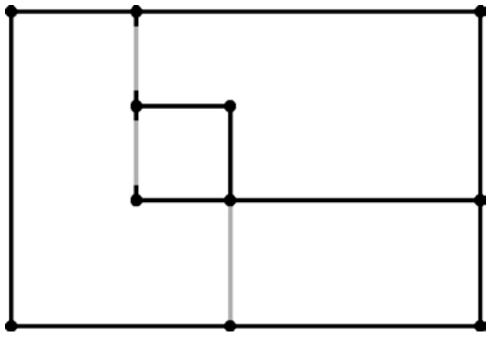


Figure 111: Two rooms with doors and a segmented, L-shaped corridor.

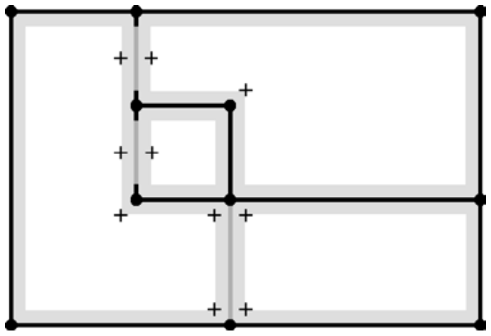


Figure 112: The constructed nodes *V* of the path network graph *G*.

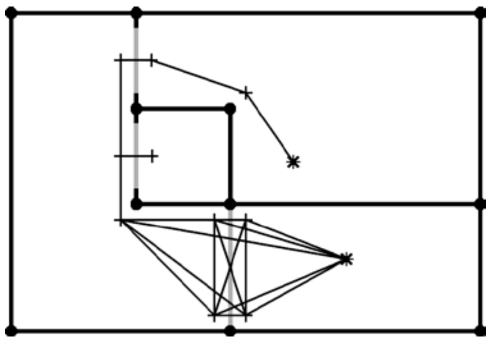


Figure 113: The constructed edges *E* of the path network graph *G*.

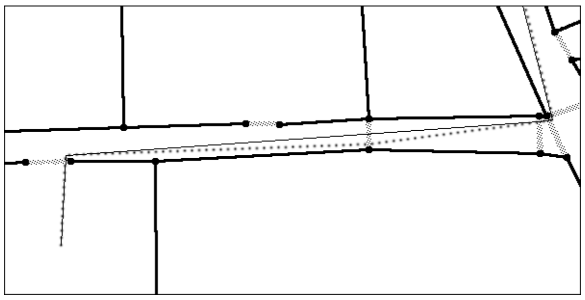


Figure 114: Optimizing a route in a segmented corridor.

7.8 Implementation

YAMAMOTO is based on the Microsoft .NET Framework[60], which is a hardware- and OS-independent platform that provides about 8000 classes for application development, including data structures, connectivity, GUIs, and multimedia. We have chosen to implement the *YAMAMOTO* editor in the C# language[61], which is easier to handle than C++ but equally efficient. The mono project[62] provides a free implementation of the .NET framework and C# for Windows, Apple Mac, and Linux. For the 3-D graphics we use OpenGL[63] technology that is provided for the .NET and mono platforms by the Tao framework[64], an open source project. The Pathfinder algorithm is exceptionally implemented in C++ so that it can be easily integrated in mobile applications for PDAs where we still use C++ for performance reasons.

The architecture of *YAMAMOTO* (see Figure 116) follows the Model-View-Controller (MVC) design pattern, which has been first formulated by Reenskaug in 1979[65] at Xerox PARC.

The **model** component represents a hybrid 2.5-dimensional model of the building's layers, regions, parametric objects, and sensors and actuators. The regions are represented as polygons that are defined over a set of vertices and connections between pairs of vertices. Connections are semantically annotated by their passableness, type, and parameters. All geometric elements can be linked to their symbolic counterparts in the UbisWorld ontology by UbisNames. The core model implements the necessary methods to create and manipulate objects and to read and write them to XML files in the YML format. The model also manages the image textures that are used for objects. The *Parametric Geometry Creation* module takes the core model and textures as input and creates the full 3-D geometry for walls, windows, doors, and objects, such as tables or shelves, according to their symbolic type and parameters. These geometries are stored as drawable objects and are used to render the building model through the *View* component.

The **view** component presents the building model to the user and facilitates interaction in a WYSIWYG (What You See Is What You Get) style; in the case of *YAMAMOTO* the view provides: i) the 3-D viewing area; ii) a grid of symbolic object properties (e.g. its UbisName, color, and passableness); and iii) the UbisWorld Ontology and the Activity Editor. The 3-D view renders the model's *drawable objects* using OpenGL and handles input events, such as mouse moves and clicks, by determining the object that the mouse is pointed at from a hidden selection buffer. It is also possible to save snapshots of the 3-D view. The Ontology and Activity Editor view is based on HTML and is rendered using Microsoft's Internet Explorer (IE) as Active-X component. Interaction between IE and *YAMAMOTO* is realized through JavaScript (JS); the .NET framework allows JS function calls from IE to C# methods.

The **controller** is responsible to inform the view component of model changes so that the UI is updated if necessary. The controller's main module implements the editor functionality. It implements a state machine that represents the current mouse mode (selecting an object, creating an object, dragging an object, etc.) and handles input events accordingly by

[60] http://msdn.microsoft.com/de-de/netframework/default.aspx
[61] http://msdn.microsoft.com/de-de/vcsharp/default.aspx
[62] http://www.mono-project.com/Main_Page
[63] http://www.opengl.org/
[64] http://taoframework.com/
[65] http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

manipulating the core model and updating the drawable objects' geometry. The controller doesn't modify the model directly, but calls the methods that are provided by the model. The animation module is used to automatically create visual route directions as movie sequences. This module comprises: i) the Pathfinder algorithm that is used to find routes between two points, ii) an interpolation algorithm that defines the frames of the movie; and iii) the avatar controls that are used to move the avatar along the route. The avatar can also be controlled interactively by the user via mouse and keyboard.

The Dual Reality interface realizes the connection between the virtual and real world. The Yamamoto Control Interface can be used by external applications to respond to changes in the editor, for example if the avatar moves, and manipulate the model through a simple set of commands. Virtual sensor and actuators directly communicate with the infrastructure of instrumented environments, such as SUPIE via the EventHeap and BAALL via the X-Gate gateway. Examples are the virtual Bluetooth proximity sensors in the SUPIE lab's model. The VNC module streams the actual content from real displays to their virtual counterparts.

7.8.1 OpenGL-Rendering Techniques

The rendering of the window's glass panes involves a translucency effect that requires a two-pass rendering strategy in OpenGL; first, all solid objects are drawn using the depth buffer, then all translucent objects are drawn over them using alpha blending. The depth buffer is set to read-only mode during the second step; otherwise windows in front would hide other windows behind them. Alternatively, all objects would have to be sorted according to their distance to the camera, and drawn beginning with the most distant objects. However, the sorting procedure would require additional processing time and can't be generally solved for all cases of overlapping objects. The depth-buffer approach is more efficient but limited to glass of the same color since the result of the alpha blending function depends on the order in which faces are drawn on top of each other.

The selection of objects from the 3-D view requires an index of which object is visible at a given 2-D screen coordinate. We create this index by drawing all objects in false colors and without shading to an invisible frame buffer (see Figure 115) and using a dictionary of colors and objects. By doing so, we can infer the object at a given screen coordinate by reading the pixel color (a 24-bit integer value) from our selection buffer. Then we use the dictionary to lookup the object by its color. The selection buffer image and color index could also be used by vision-based systems in order to identify objects in areas of blob-motion in a video stream that is taken from the same perspective as the virtual scene.

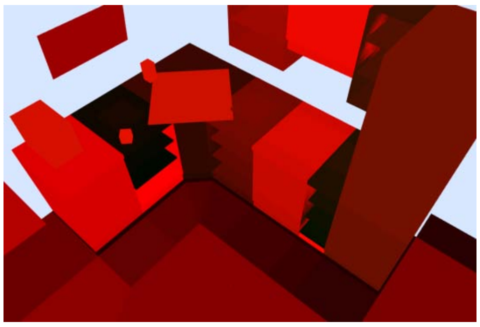


Figure 115: A false-color image is used as an index for object selection in the 3-D view.

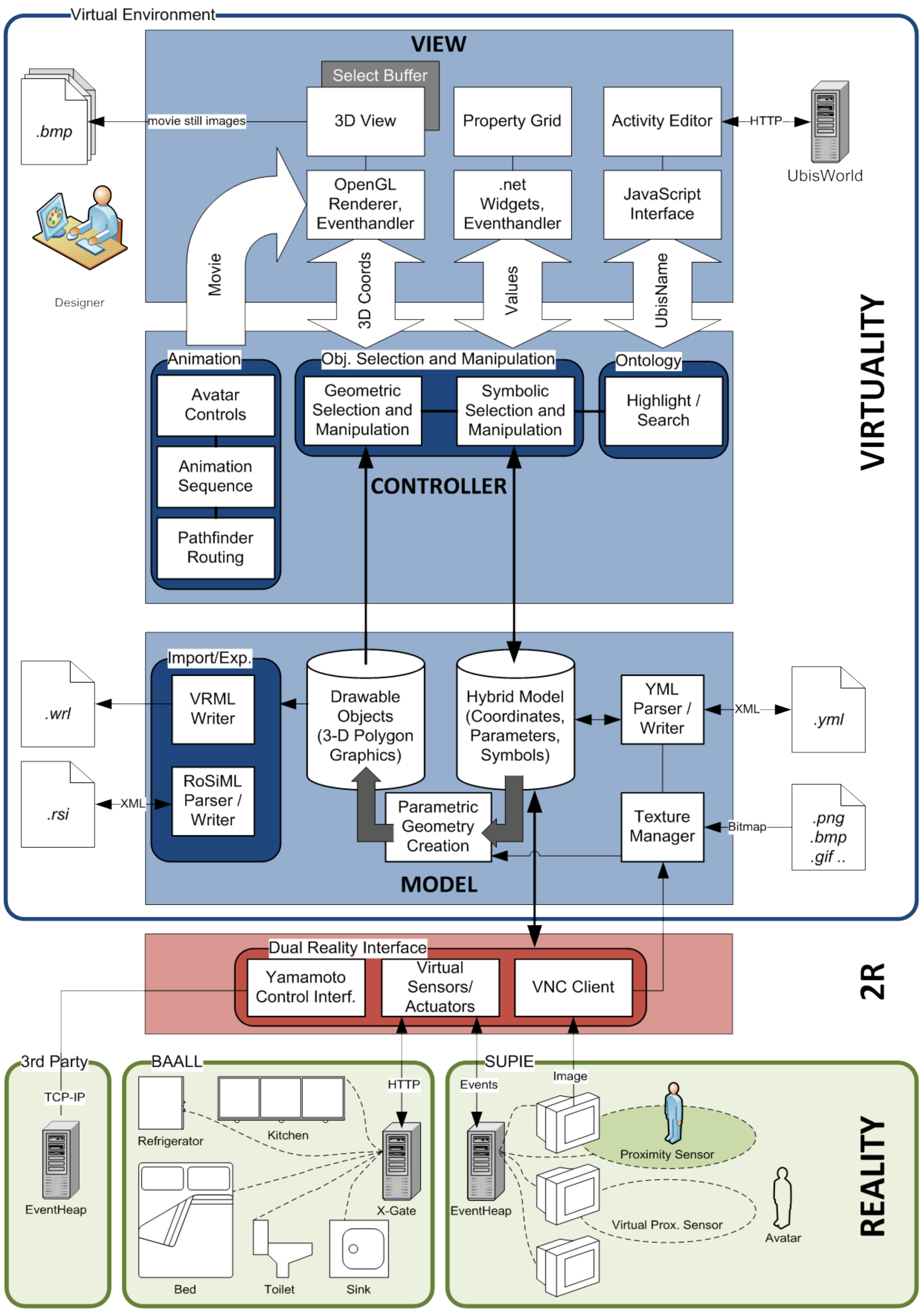


Figure 116: The system architecture of *YAMAMOTO* follows the Model-View-Controller pattern.

7.9 Summary

In the beginning of this chapter we have motivated the development of yet another map modeling toolkit (*YAMAMOTO*) with the special requirements of pedestrian navigation in built environments. In this context it is particularly important to support route finding on model multiple levels, and to have a polygonal representation instead of a route graph. As solution to these requirements, we have introduced a 2.5-dimensional representation of the building's geometry that is supplemented with semantic annotation of the accessibility of regions and the passableness of edges for route finding purposes.

Following the conceptual background, we have described the usage of the *YAMAMOTO* toolkit on a practical level, in the sense of a user manual. After explaining the components of its graphical user interface, we gave step-by-step instructions on how to create a new model; starting with a floor plan as backdrop image, adding a scale, and viewing the model from various perspectives. Then we have given a small tutorial on how to create new polygon objects in order to represent regions (e.g. rooms) of a building in 2.5D, and how to select and manipulate polygons. In this context, we also explained how to model multi-level buildings using layers, and how to model stairs that connect layers.

Next, we have introduced the use of parametric objects as a simplified way of 3-D modeling. First we presented the available elements for the building structure, such as walls, windows, and doors, and how to adapt them by their parameters, like ceiling height or frame color. Then we showed how to furnish the interior of a building with generic objects, which comprise simple boxes, textured boxes, storage elements, and tables. Furthermore, we pointed out how *YAMAMOTO* supports the modeling and design of instrumented environments; particularly how to set up beacons for indoor positioning, and how to model optical devices, such as public displays, projectors, and cameras.

In the following Section, we have introduced how hybrid models can be created that link geometric entities of the environment model with their symbolic counterparts from the UbisWorld ontology. In this context, we have explained how to assign UbisNames to spatial regions and parametric objects by using the ontology view, and how to browse and search. Based thereupon, we have introduced how *YAMAMOTO* can be applied to support the *GOAL* design method through the integrated ontology-based Activity Editor component. We have explained how to use the Activity Editor's GUI to hierarchically add and edit activities, actions, and operations, and how to drag-and-drop ontological concepts from UbisWorld.

Regarding Dual Reality scenarios, we have presented three specific 2R features: virtual Bluetooth proximity sensors, the GPS receiver emulation, and VNC display connections. Additionally we have documented an external interface that grants 3rd party applications access and control over all model elements, e.g. to move the avatar by a positioning system.

Finally, we have explained hierarchical modeling at multiple level of details, and how to derive geo-coordinates for indoor models from outdoor maps. Furthermore, we described how the PathFinder algorithm applies an A* search to find routes in a 2.5D building model.

We have closed the chapter with an overview of *YAMAMOTO's* system architecture. that follows a Model-View-Controller design pattern and highlighted the Dual-Reality interface that connects the virtual world model with the real BAALL and SUPIE environments.

8 Use Cases

In the preceding chapters, we have introduced the *GOAL* method and various features of the *YAMAMOTO* toolkit, which range from location modeling to activity modeling and route finding. In this chapter, we will highlight how we have practically applied *GOAL* and *YAMAMOTO* in five use cases, covering development aspects from design to evaluation.

We will start with the modeling and designing of the *Smart Kitchen* environment, where we have applied the *GOAL* method to model and document the environment, including its storage elements, appliances, and instrumentation with RFID antennas and a display. We will further present an activity model for the preparation of Italian bruschetta bread.

In the second example we introduce the activity-adaptive indoor navigation system *VISTO* that suggests pedestrians local destinations based on their scheduled tasks. Furthermore we describe the automated generation of visual route directions, and how we have simulated *VISTO* in a Dual Reality environment. We also present the results of a user study concerning individual differences in wayfinding performance between map and movie presentations.

The modeling of the BAALL living-lab is a representative use case from the AAL domain. We will describe how we have physically simulated and evaluated the furnishing of the lab prior to its deployment in virtual reality, focusing on the maneuverability of a wheelchair.

Furthermore, we report on how the IRL research lab has applied *YAMAMOTO* and the Path-Finder routing algorithm for the development of navigational assistance in a supermarket.

Finally, the Roaring Navigator audio-guide project highlights the advantages of *YAMAMOTO* for the modeling and simulation of a zoo environment.

8.1 Modeling User Assistance in the Smart Kitchen Environment

The kitchen has recently been discovered as an interesting application domain for AI and Ubiquitous Computing projects due to the complex and social nature of cooking-related tasks, and various assistants have been developed, like (Makino et al., 2009) for example. The *Smart Kitchen* is an instrumented kitchen environment that is situated in the DFKI office building next to the researchers' offices. The kitchen room is open to the staff and frequently used for coffee breaks and to prepare simple meals. In terms of the *SharedLife* project (Wahlster, 2006) the kitchen has been upgraded and instrumented with various sensors and a large display in order to realize the *Semantic Cookbook* (Schneider, 2007), an electronic cookbook application that provides proactive user assistance on how to prepare a meal.

This environment is an ideal test bed for us to apply and further develop the *GOAL* method (see Chapter 6) and the *YAMAMOTO* toolkit (see Chapter 7). In this section, we will present the kitchen's geometric model, the according extensions to the ontology, and the activity model that we have created. We also discuss the applicability of the models during the design and implementation phase of the project.

For example, by analyzing the activity model we have identified the need to locate artifacts and cooking ingredients as elementary part of most actions in the kitchen. So we decided to contribute short movie sequences to the system implementation that highlight shelf positions as a kind of micro-navigational aid, which helps non-frequent users of the kitchen to find the required artifacts in the cupboards. The movies have been automatically created from the geometric model.

The *Semantic Cookbook* has also been evaluated in a user study where the subjects were asked to prepare traditional Italian bruschetta, according to the advice of the system. Hence we have focused our activity model on this recipe. From the modeling of the interaction between the user and the system we have found and documented some minor bugs in the interaction with the system, which will contribute to future improvements. Finally, our models provide a useful documentation of the instrumentation with technology, since the RFID technology for example is invisible to the user.

8.1.1 Smart Kitchen Overview

The *Smart Kitchen* assistant captures the user's context and actions during the cooking as video and sensor data from the instrumented environment. For this purpose, the system is based on an instrumented environment equipped with various sensors, cameras, and networked appliances, which capture the user's actions and context during cooking. The system uses four different kinds of sensors: i) multiple wide-angle video cameras, ii) radio frequency identification (RFID) equipment, iii) an electronic scale, and iv) networked kitchen appliances. *Multiple wide-angle cameras* record an audio/video stream of the cooking session. A motion detection algorithm automatically switches to the camera that currently captures the most activity. *RFID equipment* identifies and locates ingredients and objects. Each movable object in the kitchen is tagged with a passive RFID transponder. Various *RFID antennas* are mounted discretely at important locations in the kitchen and allow detecting if an ingredient or tool is placed on or removed from such a location. The *electronic scale* measures the amount of used ingredients. Finally, *networked kitchen appliances* (stove, oven, and fridge) report about their state and usage. The stove monitors the power level of each individual hot plate and the oven provides the operation mode settings chosen by the user as well as the current temperature.

The system initially stores a stream of these raw sensor data. Then, an abstraction process is applied to create a symbolic representation of the observed information. The symbolic representation is stored as a context log. Additionally, related observations are combined into entries. With the abstraction process, a user can share the sensor data in a way that is easy to understand and handle.

The *Smart Kitchen* is equipped with a wide-screen TFT panel which is used to display the expert's cooking process captured by sensor and video. The interface is divided into three parts. The video part on the left side displays the video stream as it is captured by the cameras. For each camera, there is a small preview window in the bottom row. The object and ingredients list displayed in the middle of the screen shows the objects and food items involved in the cooking process. The information view covers the right third of the screen. Besides the recipe name, additional information about the object that is used in the current step of cooking is displayed. This could include for instance directions on how to use a certain kitchen tool, product information, or a concrete warning.

8.1.2 Geometric Modeling

According to our proposed *GOAL* method, we have started the design process with the geometric modeling of the kitchen environment. After measuring the size of the room, we have drawn a simple floor plan with Microsoft's Powerpoint software and exported it as a bitmap graphic (see Figure 117) so that 250 pixels represent one meter in reality. We have also downloaded image textures for the kitchen appliances from the vendor's Website[66]. For instance, see the front of the oven in Figure 118. In the next step we have used the Yamamoto toolkit to construct the walls and windows of the room according to the floor plan, as shown in Figure 119. We have split the floor in several spatial regions that implicitly result from the arrangement of the appliances. For example, the dishwasher and oven can only be operated from a standpoint in their front. Hence it makes sense to model these interaction zones separately so that the spatial context can be used for activity recognition. However, the spatial context is currently not being used by the *Semantic Cookbook* implementation. By setting up the model scale accordingly (1m:250px), we were able define the width and height of the kitchen furniture and appliances in centimeters. Figure 120 gives an idea of the kitchen elements' assembly in *YAMAMOTO*. The oven is modeled as a parametric box object that measures 60cm x 60cm x 78cm (HxWxD) and has its image mapped to the front, see Table 9 for the actual YML code snippet and Appendix B for the XML document type definition (DTD). The cupboard is composed of a stand that is holding a container element on the top of it which includes three boards and is covered by a worktop and RFID antenna. All objects are modeled in *YAMAMOTO* as box elements, except of the container that is represented by a storage object and the display. The according YML code is given in Table 10. The rest of the kitchen has been modeled similarly to the examples shown.

Table 9: The YML code for the oven appliance.

```
<Box name="N..987699" type="Box" x="1273" y="-10" z="59"
orientation="0" width="60 cm" height="60 cm" depth="78 cm"
color="#FAFAFA" textureFront=".\oven.jpg" textureTop="" />
```

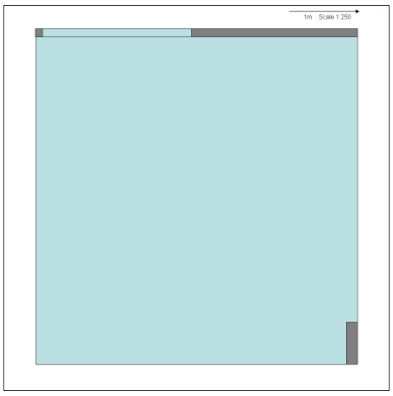


Figure 117: Kitchen floor plan, 1m = 250 px.

Figure 118: Oven image texture.
Source: Siemens-Electrogeräte GmbH.

[66] http://www.siemens-home.com/

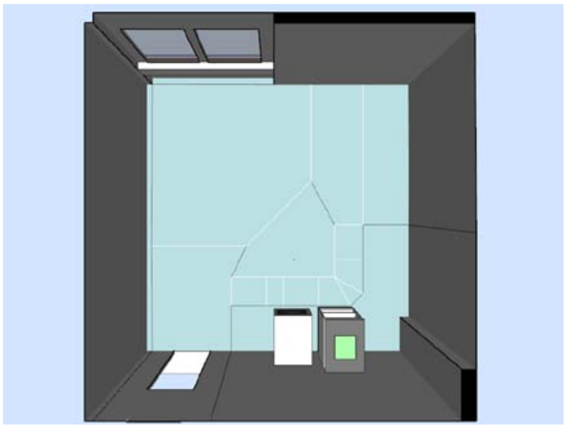


Figure 119: Walls, windows, and the door.

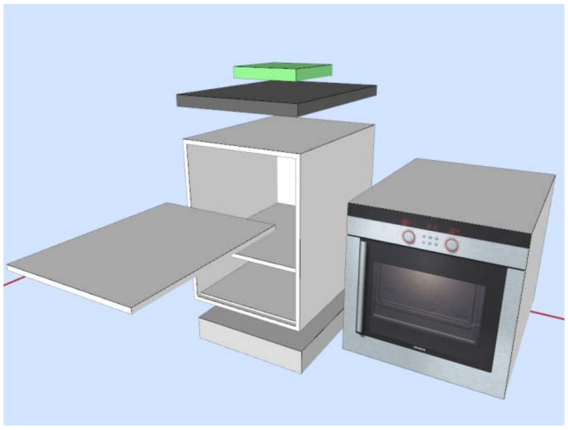


Figure 120: Assembly of the cupboard.

Table 10: The YML code for the cupboard.

```
<Box name="_SchrankGruppe2" type="Group" x="698" y="876" z="0"
orientation="0" width="60cm" height="80cm" depth="78cm" color="#FAFAFA">
   <Box name="_Sockel" type="Box" x="0" y="3,723" z="0" orientation="0"
width="60 cm" height="12 cm" depth="76 cm" color="#D3D3D3"/>
   <Box name="N..987686.D31_2DC016_2DSchrank2" type="Storage, t=2cm" x="0
" y="1,861818 " z="54" orientation="0" width="60cm" height="80 cm"
depth="78cm" color="#FAFAFA" >
      <Box name="N..987672.D31_2DC016_2DFach2_2D1" type="Box" x="0" y="0"
z="1,861818" orientation="0" width="55,5cm" height="2cm" depth="73,5cm"
color="#FAFAFA" />
      <Box name="N..987673.D31_2DC016_2DFach2_2D2" type="Box" x="0" y="0"
z="50,26909" orientation="0" width="55,5cm" height="2cm" depth="73,5 cm"
color="#FAFAFA" />
      <Box name="N..987674.D31_2DC016_2DFach2_2D3" type="Box" x="0" y="-
170" z="98,67636" orientation="0" width="55,5cm" height="2cm" depth="73,5
cm" color="#FAFAFA" />
   </Box>
   <Box name="Platte3" type="Box" x="0,3106079" y="0,2896729"
z="230,2873" orientation="0" width="60cm" height="5cm" depth="80cm"
color="#696969"/>
   <Box name="N..987837.antenna2" type="Box" x="0 " y="0" z="176"
orientation="0" width="32cm" height="5cm" depth="34cm" color="#90EE90"/>
</Box>
```

8.1.3 Extending the Ontology

After finishing the geometric model, we have extended the *UbisWorld* ontology (Heckmann, 2006) with all concepts and instances that are necessary to represent the kitchen domain. Particularly the regions and objects of the geometric model have to be represented. In the spatial branch of the ontology we have added a room instance for the kitchen itself and its regions. Their topological inclusion is expressed as a *"Part-Of"* relation in the *UbisEarth* tree (2.4.4.2). As a specialty, we have extended the spatial model to the furnishing. We have modeled the cupboards and the boards inside of them in the spatial ontology and linked them to the kitchen-room instance, again using the *"Part-Of"* relation. This approach allows us to express the location of tools and ingredients in the kitchen's cupboards by situational statements that refer to unique identifiers (UbisNames). These location descriptions can also be used to visually localize things in the model, as we will see later.

Regarding the artifacts, we have extended the physical branch of the ontology with the necessary concepts for kitchen appliances. We have added classes and instances for the dishwasher, oven, cooking plate, refrigerator, freezer, and the espresso machine.

Finally we have added the instrumentation of the environment, which comprises of the 30 inch TFT display, three RFID antennas, and the control cube which has been designed to control the playback of the Semantic Cookbook.

Figure 121 shows the resulting kitchen model that has been rendered with *YAMAMOTO*. We have manually added some snippets from the UbisWorld ontology to illustrate the strong relationship between geometry and ontology.

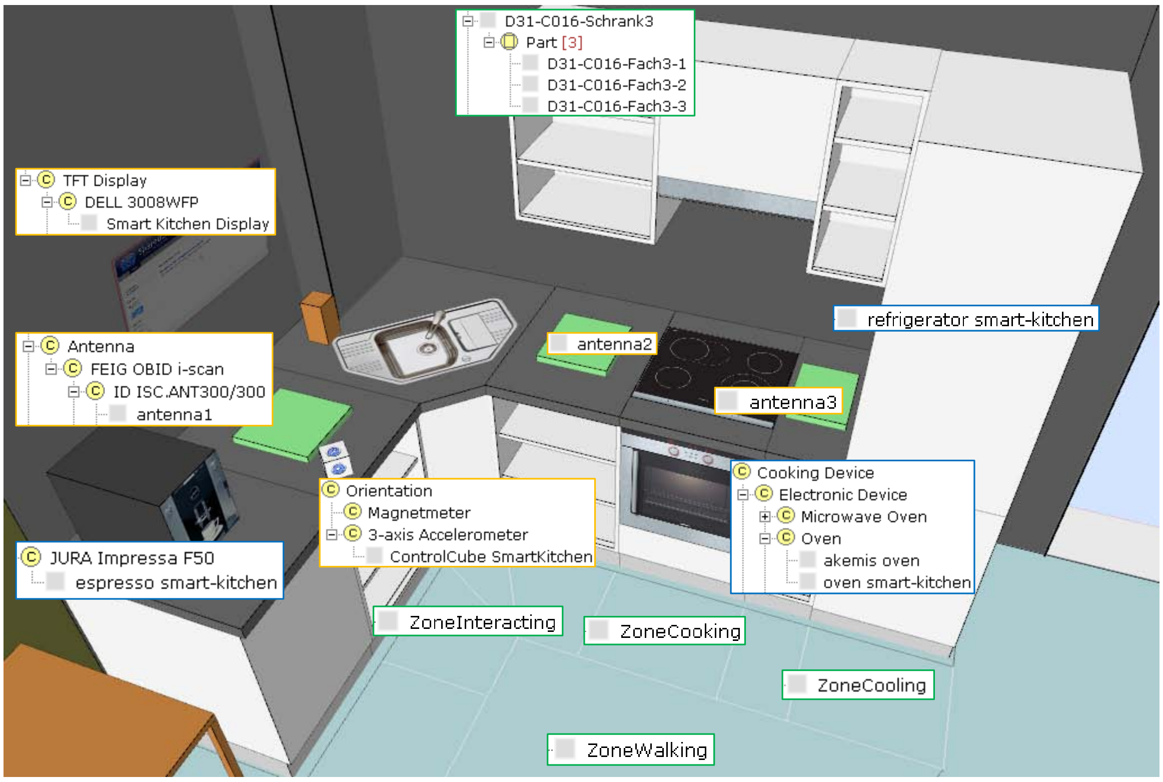


Figure 121: Relationship between the geometric model and the ontology.

8.1.4 Activity Modeling

Considering activities of daily living, the kitchen environment is one of the most interesting places at home with respect to the complexity of activities and actions that are performed, and the manifold of artifacts that are used. Hence the *Smart Kitchen* scenario is a very promising use case for the *YAMAMOTO* activity editor.

In order to evaluate the *Semantic Cookbook's* usability, a user study has been conducted in the smart kitchen. The preparation of Italian bruschetta bread has been chosen as task since it is a relatively simple recipe that can be performed with moderate effort even by inexperienced subjects. We have used the *YAMAMOTO* activity editor to model the necessary actions and operations related to this recipe, with respect to artifacts and spatial zones in the smart kitchen.

We have modeled the preparation of bruschetta with and without user assistance as two different activities. The first activity just describes typical actions that are related to the preparing of the bruschetta, whereas the second activity includes the interaction between the user and the system.

It should be emphasized that the activity model only lists the various actions and operations once in order to represent their relationship to artifacts, locations, and the instrumentation of the environment. The model is not intended to represent plans, for example step-by-step instructions on how to prepare bruschetta, since the tabular form is not able to properly represent repetitions or decision points or branches. This is because our work focuses on the design phase, in particular to help the designer to decide which actions require assistance, and how to let the environment sense user activity. We will discuss later how the activity model can be used during the implementation phase for planning or plan recognition.

We will now describe in full detail two exemplary actions (see Table 11 for the abridged activity table) from the non-assisted bruschetta activity. The first example action is named "Activate Oven" and subsumes the operations that are necessary to set up the oven's heat type and temperature in order to toast the bread. Hence the action and operations refer to the oven as artifact and to the zone in front of the oven. Both can be directly visualized in the geometric model as shown in the *YAMAMOTO* screenshot in Figure 122. The according *UbisWorld* identifiers are referred to by the footnotes, e.g. the role of the user is represented by the UbisName *N..00800480* . The second example describes the preparation of tomatoes. It includes the operation to take the tomatoes from their storage, typically the fridge. The corresponding location is the zone in front of the fridge-freezer combination. After the tomatoes have been taken, they need to be cut and sliced into small cubes. Therefore the knife and cutting board artifacts are used.

Table 11: Exemplary actions

Activity / Action / Operation	**Subject**	**Artifact**	**Purpose**	**Location**	..
Prepare Bruschetta as starter					
Activate Oven	user[1]	oven[2]		ZoneCooking[3]	
switch on	user[1]	oven[2]	Activate oven	ZoneCooking[3]	
choose heat type	user[1]	oven[2]	Set proper heat type to toast bread	ZoneCooking[3]	
set temperature	user[1]	oven[2]	Set temperature to 180° C	ZoneCooking[3]	
Prepare Tomato					
Take tomato from storage	user[1]	fridge[4]		ZoneCooling[5]	
Cut tomato in small cubes	user[1]	Knife[6], cutting board[7]	Prepare tomato cubes	ZonePreparing[8]	
..	..	..	..	.	..

UbisNames: [1)]*N..00800480*, [2)]*N..00987699*, [3)]*N..00987818*, [4)]*N..00987143*, [5)]*N..00987816*, [6)]*N..00987874*, [7)]*N..00987882*, [8)]*N..00987819*

In the second version of the bruschetta activity we have also modeled the interaction between the user and the assistance system. The *Semantic Cookbook* has been designed to instruct the user how to prepare a selected dish step-by-step by presenting multimedia content of pre-recorded video and speech. At the beginning of each step, the system gives a verbose summary of what comes next and the display lists the required artifacts and ingredients. The user now has to put the listed items on the kitchen's worktop where they will be sensed through the RFID reader. If all items are complete, the system proactively continues the playback of the instructions. Alternatively, in case that there are problems with the RFID tags or readers, the user can manually confirm that the necessary items are at hand. We will now see how we have modeled this schema using the activity editor.

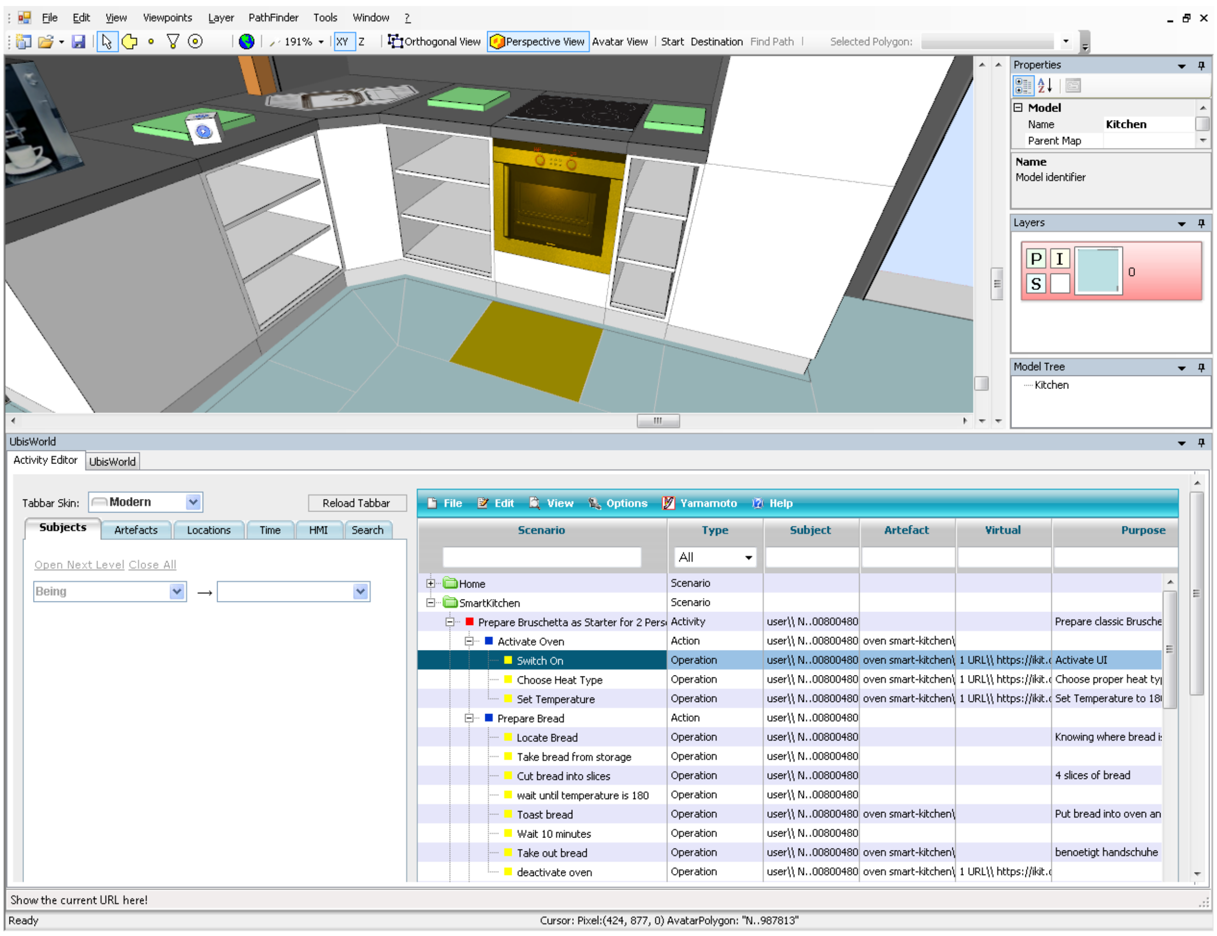
Table 12 exemplary shows an excerpt of the activity model for the first section of the recipe. This section instructs the user how to cut the bread into slices.

Beginning with the first breakpoint, the system lists the cutting board, bread knife, and bread on the display, then plays a gong sound, and a voice instructs the user of what will come in the next section: "*Bread: cut the bread into approximately 8mm thick slices*". We have modeled this situation as a system-initiated action (note that the subject refers to the *Semantic Cookbook*) with three presentation operations, where the HEI (Human-Environment Interaction) attribute refers to the TFT-display in the kitchen.

The system now waits for the confirmation of the user. This step is modeled as a user-initiated action with two operations; these are the manual conformation with the control cube and the proactive confirmation with the RFID antenna and reader. The latter operation also refers to the required artifacts of bread knife, bread knife, and bread.

Table 12: Assisted preparation of bruschetta

Activity / Action / Operation	**Subject**	**Artifact**	**HEI**	..
Prepare Bruschetta (assisted)				
Breakpoint1	SemanticCookbook^{1}			
Gong	SemanticCookbook^{1}		Display	
Text and speech	SemanticCookbook^{1}		Display	
Highlight items	SemanticCookbook^{1}		Display	
Confirmation1	user^{1}			
Manual confirmation	user^{1}		Control Cube^{4}	
Place cutting board, bread knife, bread on RFID area	user^{1}	Bread-knife^{6}, cutting board^{7}, bread	Antenna1	
Present section 1	SemanticCookbook^{1}			
Play video section 1	SemanticCookbook^{1}		Display	
Perform section 1	user^{1}			
Cut bread in slices	user^{1}	Bread-knife^{6}, cutting board^{7}, bread		
..	..	..	..	..



Scenario	Type	Subject	Artefact	Virtual	Purpose
Home	Scenario				
SmartKitchen	Scenario				
Prepare Bruschetta as Starter for 2 Pers	Activity	user\\ N..00800480			Prepare classic Brusche
Activate Oven	Action	user\\ N..00800480	oven smart-kitchen\		
Switch On	Operation	user\\ N..00800480	oven smart-kitchen\	1 URL\\ https://ikit.	Activate UI
Choose Heat Type	Operation	user\\ N..00800480	oven smart-kitchen\	1 URL\\ https://ikit.	Choose proper heat ty
Set Temperature	Operation	user\\ N..00800480	oven smart-kitchen\	1 URL\\ https://ikit.	Set Temperature to 18
Prepare Bread	Action	user\\ N..00800480			
Locate Bread	Operation	user\\ N..00800480			Knowing where bread i
Take bread from storage	Operation	user\\ N..00800480			
Cut bread into slices	Operation	user\\ N..00800480			4 slices of bread
wait until temperature is 180	Operation	user\\ N..00800480			
Toast bread	Operation	user\\ N..00800480	oven smart-kitchen\		Put bread into oven an
Wait 10 minutes	Operation	user\\ N..00800480			
Take out bread	Operation	user\\ N..00800480	oven smart-kitchen\		benoetigt handschuhe
deactivate oven	Operation	user\\ N..00800480	oven smart-kitchen\	1 URL\\ https://ikit.	

Show the current URL here!

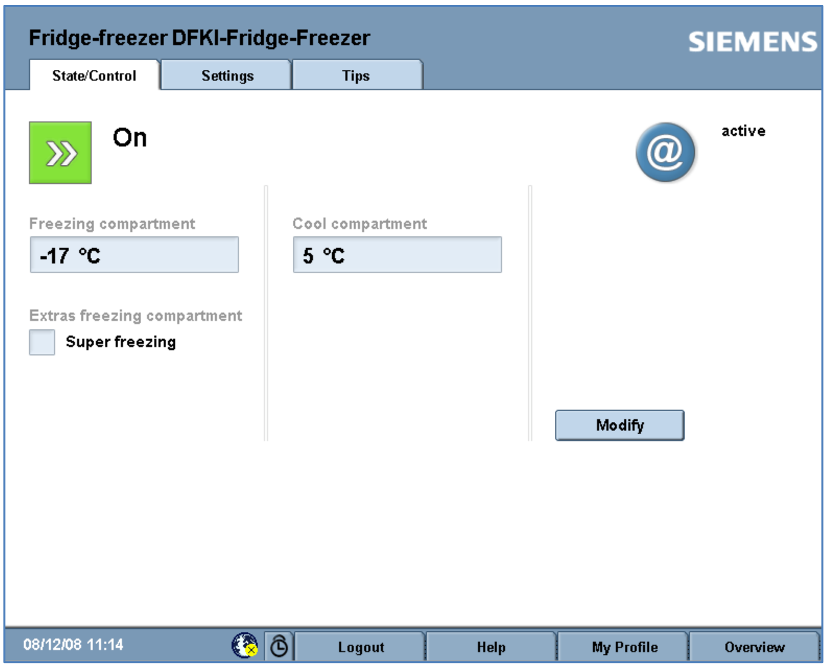
Ready

Cursor: Pixel:(424, 877, 0) AvatarPolygon: "N..987813"

Figure 122: Visualizing the oven artifact and location from where the oven is operated.

In the next step the system plays the video section that shows how the bread is cut. This is again modeled as the action "present section 1" with a single operation, where the subject refers to the Semantic Cookbook and the HEI refers to the display. Accordingly, the part of the user is modeled as the action "perform section 1", where the subject again refers to the role of the user, and the artifacts refer to the board, knife, and bread. This interaction schema is repeated in each section until the recipe is complete.

In this context, two virtual artifacts are worthwhile to mention. The oven appliance is connected to the Web via the *serve@Home* framework, so that the oven's physical user interface has a digital counterpart online that we can refer to with its URL. The dishwasher and fridge appliances are also available on the Web, see Figure 123 for example. Likewise, the multimedia content of the *Semantic Cookbook* can be considered as virtual artifact.


Fridge-freezer DFKI-Fridge-Freezer

SIEMENS

On

active

Freezing compartment: -17 °C

Cool compartment: 5 °C

Extras freezing compartment

Super freezing

08/12/08 11:14

Figure 123: Fridge-freezer user interface on the Web.

8.1.5 Micro-navigational Assistance

Navigational tasks are usually associated with wayfinding problems in buildings, for example how to locate the smart kitchen room at the DFKI. However, once the user has found and entered the kitchen, the localization of artifacts and ingredients poses a similar problem. We conceive this as a micro-navigational task in contrast to macro-navigation, as described in Section 2.3.3.

In order to assist the users in their micro-navigational tasks in the kitchen, we use the three-dimensional model of the kitchen in order to visualize the shelf board that contains the desired item. Therefore we have used the *YAMAMOTO* toolkit to prepare short animations that start with the perspective of the user looking at the screen, followed by a pan and zoom sequence that focuses on the respective shelf board. Additionally, the board is highlighted in orange color. For example, assume that the bread is located in the lower part (board 3) of the upper left cupboard (shelf 3) in the *Smart Kitchen*. The according animation is depicted as series of still images in Figure 124. The animations have been integrated into the Semantic Cookbook user interface that is shown on the TFT display in the kitchen and appear in a small window next to the listed artifacts and ingredients.

Other solutions could have been to instrument the shelf with flashing lights in order to guide the user, or to use a steerable projector in order to project information on the shelf.

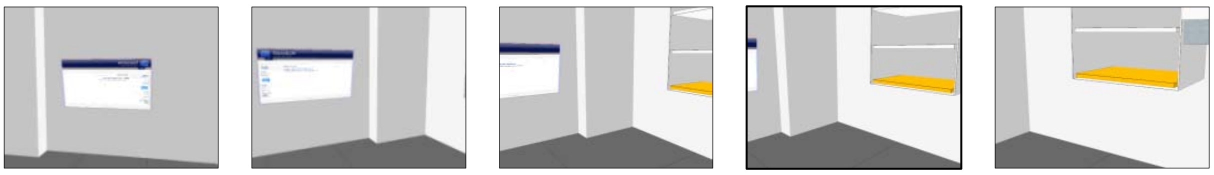


Figure 124: Image sequence from the movie that focuses on the lower board in the cupboard.

8.2 Activity-Based Indoor Navigation Assistance

VISTO (**VI**deos for **S**pa**T**ial **O**rientation) has been designed to proactively support navigational tasks in indoor environments by presenting animated route descriptions. In (Stahl et al., 2009) we have described how the activity-adaptive interface seamlessly connects with our own, Web-based organizer application called *UBIDOO* (**UBI**quitous To-**DO** **O**rganizer) in order to adapt the destination selection process to the scheduled activities of the user. We will begin this section with a detailed description of *VISTO's* functionality and user interface, before we discuss the difficulties related to the automatic generation of visual route descriptions. We will also present how we applied the *YAMAMOTO* toolkit to pre-compute a complete set of route descriptions within a building, so that no route computations are necessary at runtime. Finally we describe how the simulation capabilities of *YAMAMOTO* helped us to design and evaluate the system prior to its deployment. *VISTO* and *UBIDOO* have been developed in the project *RENA* (*"Resource-Adaptive Navigation"*, 2005-2007, Transfer Unit 53, Funded by DFG), which followed the research in the project REAL, which has been described in Chapter 5.

8.2.1 An Activity-Adaptive User Interface

In some public places you can already find information kiosk systems that present basic information about local services and maps for orientation. However, these systems are not frequently used even though they provide interesting information for the user. We believe that the reason for that are the many steps of interaction until the desired information can be retrieved through the user interface. Typically, the user has to begin the dialogue with some basic choices, such as language, and navigate through many sub-menus. Often it feels easier to ask people or to consult traditional maps, especially in situations of time pressure. For instance, if we arrive at the airport and have to locate our check-in desk, we would not like to experiment with some machine and try to figure out how it works; it is often not even clear what kind of knowledge we can expect from these systems.

We strongly believe though that public or ambient displays, if properly designed, have several advantages over personal devices. First, the diversity of mobile devices regarding their operating systems, technical capabilities, and form factors makes it hard to implement assistance applications that run on any PDA or mobile phone. Secondly, all kinds of connectivity issues arise today, like wireless LAN access keys or different GSM frequency bands worldwide. Finally, mobile devices would require standardized accurate positioning services like *LORIOT* in order to provide navigational aid, which will not be available in the near future. Hence we use wall-mounted public displays whose position and orientation within the built environment can be statically modeled.

We have designed *VISTO* as an Activity-Adaptive System (see Section 4.3) which is able to automatically recognize a user's identity from their mobile device through the wide-spread Bluetooth wireless protocol, as the user passes an ambient display. The trick is that no data connection is required, so the users don't have to interact with their mobile device in order to set up anything except enabling Bluetooth. Once the user is identified, *VISTO* can retrieve information about their scheduled tasks from the Web-based *UBIDOO* organizer. *UBIDOO* applies Activity Theory in a sense that the user can hierarchically structure their tasks and relate them to a set of predefined activities, which are accessible through the UbisWorld ontology. Vice versa, the spatial location model of UbisWorld allows to associate places with typical activities. The combination of both relations, *task - activity* and *activity - location*, yields in a powerful association between tasks and places. Thus *VISTO* can automatically sort out only relevant dates and tasks from the user's organizer and suggest a clear and useful set of destinations in the local area.

8.2.2 Introducing the Ubiquitous Organizer UBIDOO

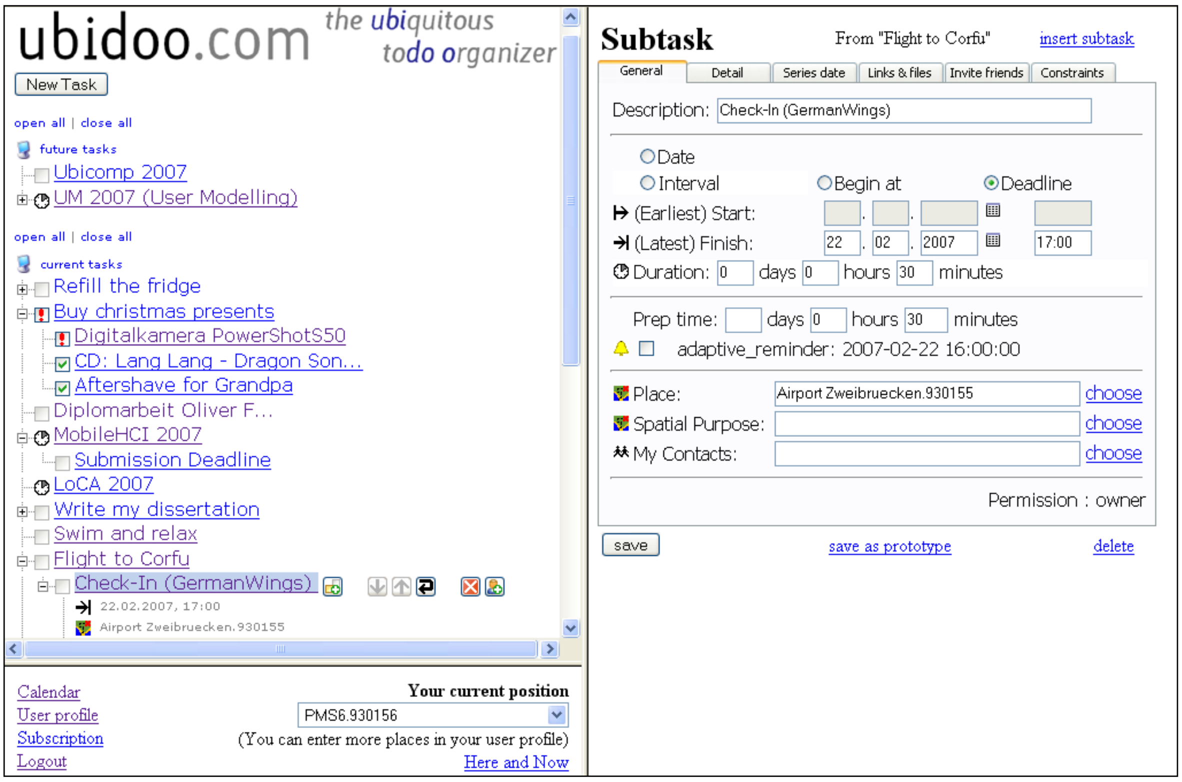
With the advent of mobile computing devices, personal information management (PIM) tools have been implemented on PDAs and are nowadays a common feature of mobile phones. Usually, the devices offer a calendar that allows to add events and to set an alarm time for a reminder. The calendar is typically supplemented by a to-do list or notepad, where the user can add tasks that have no certain date. Whereas the conceptual model behind these PIM tools is well understood by both the designers and the users, it has obvious drawbacks that we want to address in this section.

Our ubiquitous organizer *UBIDOO* integrates the conceptual models of a diary and a to-do list into a hierarchical task tree, where each task can have multiple subtasks. Each task acts a reminder for the user to do a certain activity (or action) at a certain time and at a certain place. The subtasks can either represent alternative choices, multiple goals, or fixed sequences of actions. Similar to a to-do list, the status of each task is graphically represented by a checkbox. Initially, the task is not assigned and the checkbox is disabled. Once the task is assigned to someone, the checkbox becomes enabled. Finally, the task can be checked by the responsible user to indicate completion or marked with a warning sign to indicate failure. In order to support coordination, multiple users can be invited and associated with tasks. Collaborative tasks are also supported as documents, and references to websites can be attached to shared tasks.

The *UBIDOO* task planner is running on a Web-server[67] and can be accessed via three different user interfaces that have been optimized for the following situations: At home or at work, in a mobile context, and inside of a built environment. The preferred place to maintain the organizer and to add new tasks would be at home or at work, where a desktop PC with keyboard and large screen is available. The user interface for desktop PCs is split into three frames, as shown in Figure 125. On the left hand side, the users' tasks and subtasks are shown in a foldable tree structure. By selecting a task from the tree on the left, a summary of its details (dates, location and description) is shown and then icons allow moving or deleting this task. In the right frame, various properties of the selected task can be edited and be arranged into six tabs. The bottom frame provides links to the user profile and a traditional calendar view. The user can also manually choose a location for an adapted task view ("here and now") that will be described later in more detail.

For the usage of the organizer in a mobile context, a reduced interface without frames and tabs is also available and can be accessed through any mobile device that provides a Web browser with Javascript support. We have optimized the layout for Nokia *E60* mobile phones. In addition, we have implemented two features that adapt the reminder function and the graphical presentation according to the current time and place of the user.

[67] The **ubi**quitous to-**do o**rganizer. Website: http://www.ubidoo.com/


ubidoo.com the ubiquitous todo organizer

New Task

open all | close all

future tasks
- Ubicomp 2007
- UM 2007 (User Modelling)

open all | close all

current tasks
- Refill the fridge
- Buy christmas presents
  - Digitalkamera PowerShotS50
  - CD: Lang Lang - Dragon Son...
  - Aftershave for Grandpa
- Diplomarbeit Oliver F...
- MobileHCI 2007
  - Submission Deadline
- LoCA 2007
- Write my dissertation
- Swim and relax
- Flight to Corfu
  - Check-In (GermanWings)
    - 22.02.2007, 17:00
    - Airport Zweibruecken.930155

Calendar
User profile
Subscription
Logout

Your current position: PMS6.930156
(You can enter more places in your user profile)
Here and Now

Subtask From "Flight to Corfu" insert subtask

General | Detail | Series date | Links & files | Invite friends | Constraints

Description: Check-In (GermanWings)

Date
Interval [ ] Begin at [✓ Deadline]

(Earliest) Start: . .
(Latest) Finish: 22 . 02 . 2007 17:00
Duration: 0 days 0 hours 30 minutes

Prep time: days 0 hours 30 minutes
adaptive_reminder: 2007-02-22 16:00:00

Place: Airport Zweibruecken.930155 choose
Spatial Purpose: choose
My Contacts: choose

Permission : owner

save | save as prototype | delete

Figure 125: The Web-based user interface for the ubiquitous task planner UBIDOO.

Usually, calendars in mobile devices offer an alarm function that reminds the user on events at a fixed time and date. Setting the proper time for an alarm requires the user to consider everything that needs to be done and prepared before the event, most importantly how to go there. Some events might require the user to go home first and to dress accordingly or to pickup some things, which takes additional time. The user has to plan and foresee the whole situation under uncertainty. However, often the situation changes in an unpredictable manner and users are not always there where they planned to be. Thus the alarm will remind us too early, unnecessarily distracting us from our work, or worse, remind us too late and we can't reach the event in time.

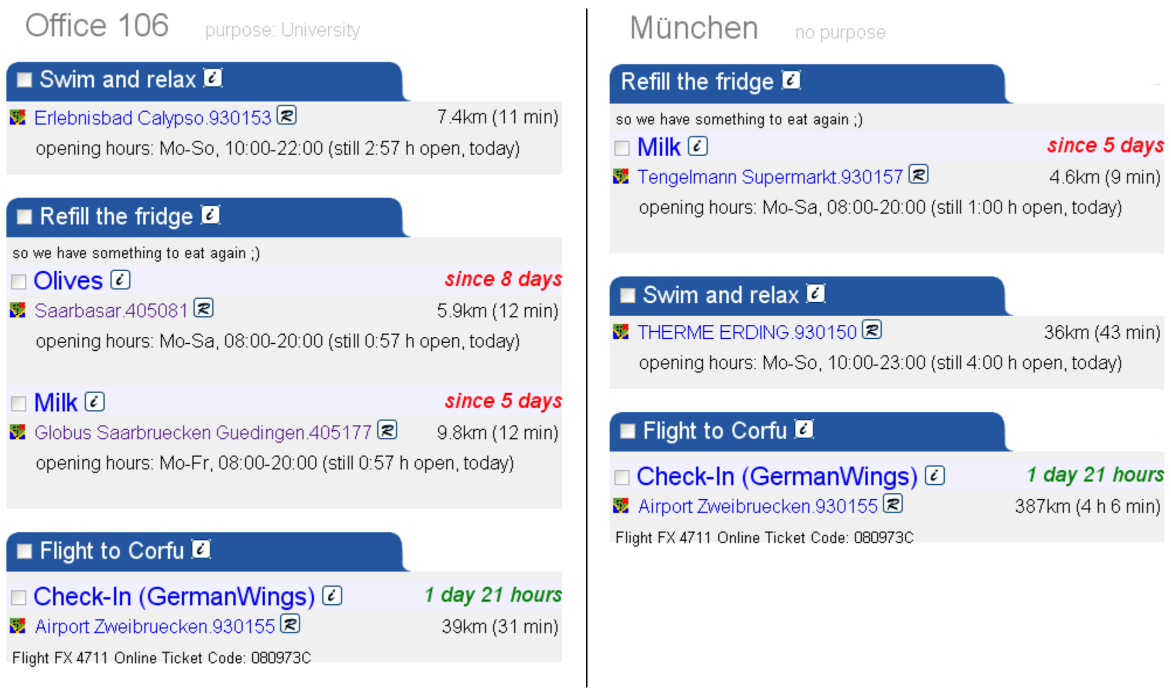
Our ubiquitous task planner addresses this issue through an adaptive reminder, which continuously recalculates the best time for the alarm based on the current geographic distance between the user and the event. In addition, a general preparation time can be specified and will be considered by the reminder. Tasks can be localized by specifying a location from the spatial ontology in *UbisWorld* (see the "Place" attribute of a task in the "General" tab in Figure 125. As the location of the user (according to the user profile) changes the task planner updates the distance between the user and all events using route planning web services. We use the *eRoute* service that is provided by PTV AG[68] and returns the estimated driving time between two locations. In the future, Web services for public transportation and pedestrians could be easily added. The planner would than choose between car and bus based on the user's preferences.

[68] http://www.ptv.de/ visited 23.07.2009

A further shortcoming of mobile organizers is their limited display size. Browsing through calendar entries and large to-do lists is a cumbersome procedure. Hence we have implemented the 'here-and-now' view (Stahl et al., 2007) in *UBIDOO*. It filters the user's tasks according to the current situation which depends on time and location of the user, so that it suggests only those tasks which can be actually performed. The time horizon of this view is limited by the next binding event to occur. Regarding location, the adaptive view does not simply match the location of the task with the user; similar to the adaptive reminder, the system performs a route calculation to estimate the actual distance and time to get there. Furthermore, the system considers the opening hours of a place, so that the planner can filter those locations which are currently closed. In combination with route knowledge, it can even find out if the location will close before the user can actually get there.

UBIDOO also considers the purpose of locations in terms of activities on a semantic level. If the user associates a task in the planner with activities from the ontology such as *shopping for electronics* instead of a certain location instance or store, the adaptive view automatically performs a search within *UbisWorld's* spatial elements and suggests the closest suitable location nearby for this task. Depending on the current time and location of the user, the system might suggest different places for the same task.

Figure 126 shows the same tasks as seen from two different locations. On the left image, we see the adapted view for Saarbrücken. The first task, swim and relax, is associated with the spatial purpose of *waterworld* and the planner suggests a place called *Calypso* in Saarbrücken that is located 7 kilometres (or 11 minutes by car) away from the current location of the user (in their office). A click on the *R* icon opens a route planner that provides a map and driving directions. The second task reminds the user to buy olives and milk. For the first item, a store called *Saarbasar* is set by the user as location, whereas for the second item the general activity *shopping.food* has been specified so that the planner automatically suggests the *Globus* supermarket.


Office 106 purpose: University

Swim and relax
Erlebnisbad Calypso.930153 — 7.4km (11 min)
opening hours: Mo-So, 10:00-22:00 (still 2:57 h open, today)

Refill the fridge
so we have something to eat again ;)
Olives — since 8 days
Saarbasar.405081 — 5.9km (12 min)
opening hours: Mo-Sa, 08:00-20:00 (still 0:57 h open, today)
Milk — since 5 days
Globus Saarbruecken Guedingen.405177 — 9.8km (12 min)
opening hours: Mo-Fr, 08:00-20:00 (still 0:57 h open, today)

Flight to Corfu
Check-In (GermanWings) — 1 day 21 hours
Airport Zweibruecken.930155 — 39km (31 min)
Flight FX 4711 Online Ticket Code: 080973C

München no purpose

Refill the fridge
so we have something to eat again ;)
Milk — since 5 days
Tengelmann Supermarkt.930157 — 4.6km (9 min)
opening hours: Mo-Sa, 08:00-20:00 (still 1:00 h open, today)

Swim and relax
THERME ERDING.930150 — 36km (43 min)
opening hours: Mo-So, 10:00-23:00 (still 4:00 h open, today)

Flight to Corfu
Check-In (GermanWings) — 1 day 21 hours
Airport Zweibruecken.930155 — 387km (4 h 6 min)
Flight FX 4711 Online Ticket Code: 080973C

Figure 126: The location-adaptive task view as seen on a mobile device in Saarbrücken (left) and Munich (right).

The last task is a reminder to catch the flight from Zweibrücken; it takes 31 minutes to go there from the office. Now compare the times and distances with the adaptive view for Munich on the right. The olives do not appear, since the store is too far away. For the milk, a different store in Munich is suggested and also the waterworld has been substituted by a local place. The adaptive reminder for the airport will happen earlier, since the estimated travelling time is now more than 4 hours.

If the user decides to assume one of the suggested tasks, the routing information can be accessed directly from the mobile phone by touching the map icon, so that the user is able to walk or drive to the new destination.

Once the user arrives and enters the built environment, *VISTO* runs on a public display infrastructure that basically provides the same "here-and-now" user-adaptive view as in the mobile situation, but enhanced with videos for spatial orientation. Again, the context-adaptive view reduces the user's cognitive load that would be required to browse through the full task tree. Furthermore, the interaction capabilities of public displays are often limited since large touch panels are quite expensive and the displays are often mounted out of reach to prevent vandalism. So it is important to avoid any login procedure and browsing. For this reason we have developed a public display infrastructure called *IPlay Blue* (Schöttle, 2006) which automatically recognizes nearby users by scanning for their mobile Bluetooth devices. This method allows us to automatically present user- and location-adaptive information to the user.

8.2.3 The User Interface of VISTO

The *VISTO* user interface is designed to assist pedestrians in indoor environments and appears on wall-mounted public situated displays, which are spread across the building at decision points. The layout is split into three areas, as shown in Figure 127. On the top border, a status bar shows the name of the location and the registered activities which are used to search and filter the *UBIDOO* task tree for. The status bar also includes a personalized icon for each user that has been currently recognized by the Bluetooth scanner. The user icons are continuously updated after each scan (each scan takes approx. 15 seconds).

On the left hand side the activities of the recognized users are shown as blue-labelled tabs. They show a list of tasks (actions), according to the hierarchic task tree structure in *UBIDOO*, which remind the user of his objectives depending on the current spatial context. In the given example, one tab represents a shopping list of products, while another tab represents a list of persons to visit. In the context of *VISTO*, each list item represents a navigational goal and provides some basic information to the user about the walking direction and distance (relative to the current display's orientation and location, which is known to the system through the Yamamoto environment model). The tasks are sorted by increasing distance, so that the closest goal appears on the top of the list. If the display affords interaction by means of a touch screen, keyboard, or mouse, the user can activate any task from the list by clicking on its arrow symbol. There can be only one active task per user. If multiple users are recognized by the system, their tasks are shown in the same list, but marked with the personal icon of their owner.

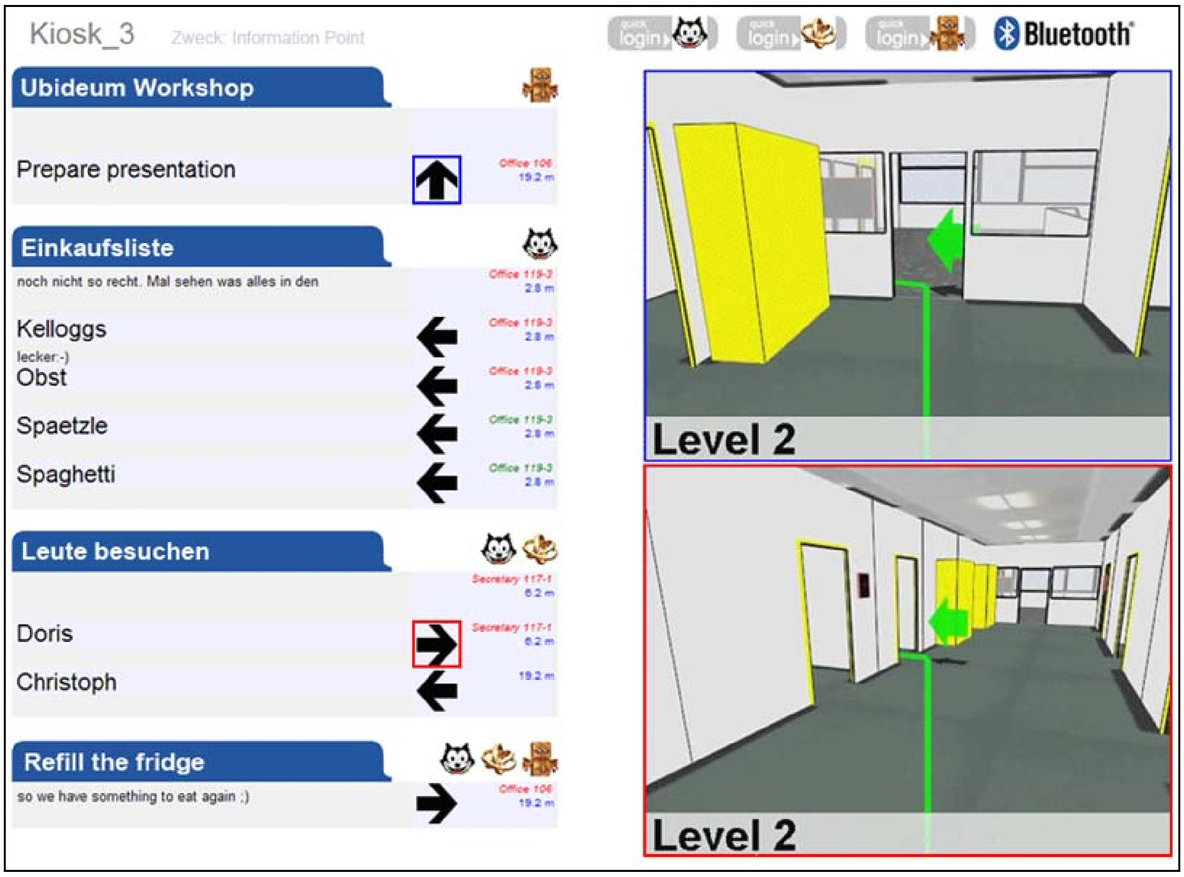


Figure 127: The activity-based user interface of VISTO.

On the right hand side, video animation frames present detailed walking directions to each active goal as a 3-D animation. In case of multiple users, colored frames (e.g. red or blue) relate the active tasks to the video frames showing their route descriptions. The active task of each user is stored in their user profile within *UbisWorld*. The *VISTO* system picks up the idea to use animated three-dimensional visualizations of indoor routes as navigational aid for pedestrians; this idea has been introduced earlier by Baus (2003) in the project *REAL*. We have re-implemented the functionality of the prior system based on the *YAMAMOTO* toolkit by using its routing and rendering engine, so that our system is able to automatically generate arbitrary indoor routes across multiple levels. In order to reduce the cognitive load of the user, routes are split into incremental route descriptions that guide the user from one display to another, whenever possible. Each animation segment shows the route from the egocentric perspective of the user. The movie starts at the current location and the avatar is facing the respective display within the virtual environment model. The avatar follows the path until the next display or the destination. The movie sequence loops until the user leaves the Bluetooth range of the display. If the user encounters a new display along the route, the playback of the next movie sequence is automatically triggered without any interaction. Finally, if the user has reached their destination, the current task will be deactivated.

As enhancement to *VISTO*, Kahl (2007) has implemented a system that uses a PDA to show the task lists and allows the user to select the active task on their personal device, acting as a remote control to the public displays.

We have also developed a mobile navigation visualization system called *HYBNAVI* (HYBrid

Navigation Visualization) (Mutafchiev, 2008) that provides assistance between *VISTO* terminals and outdoors between buildings. The calculations for routes, route descriptions, and visualization are done in real-time on the mobile device (a Windows Mobile PDA). *HYBNAVI* uses the same location model like *VISTO* and can dynamically re-calculate a new route if the user deviates too far from the planned route.

Whereas mobile phones are not yet powerful enough to render the visual route description in real time, it would be possible to download the pre-computed animations from *VISTO* and watch them on the go. The download could be realized either through the Bluetooth protocol or by sending an MMS.

8.2.4 System Architecture and Implementation

VISTO is based on the *Iplay Blue* framework (Schöttle, 2006) that has been designed to present personalized information on a network of public displays (see Figure 128). The identification of the user is realized by the wide-spread Bluetooth technology. Each Bluetooth-enabled device has a unique 48-bit identification number, similar to the MAC addresses of TCP/IP adapters. Each InfoPoint of the *Iplay Blue* framework comprises a display device and a Bluetooth device that continuously performs a scan for Bluetooth devices. All mobile devices with Bluetooth in visible mode will automatically answer to this request by sending their identification number. So if a user registers the Bluetooth ID of their mobile device in their *VISTO* user profile, the system can recognize the user. The InfoPoint periodically sends the recognized user-IDs as messages to the event heap.

The *BlueEventListener* is the core component of the framework. It listens to the events from the Bluetooth scanner and queries the user profiles in *UbisWorld* for the Bluetooth IDs. If a registered user is recognized, the *BlueEventListener* creates a URL that will display the adapted *VISTO* user interface. The URL is sent to the *Presentation Manager*, which advises the *ProMan* client that runs on the respective display device to open *VISTO* in a Web browser.

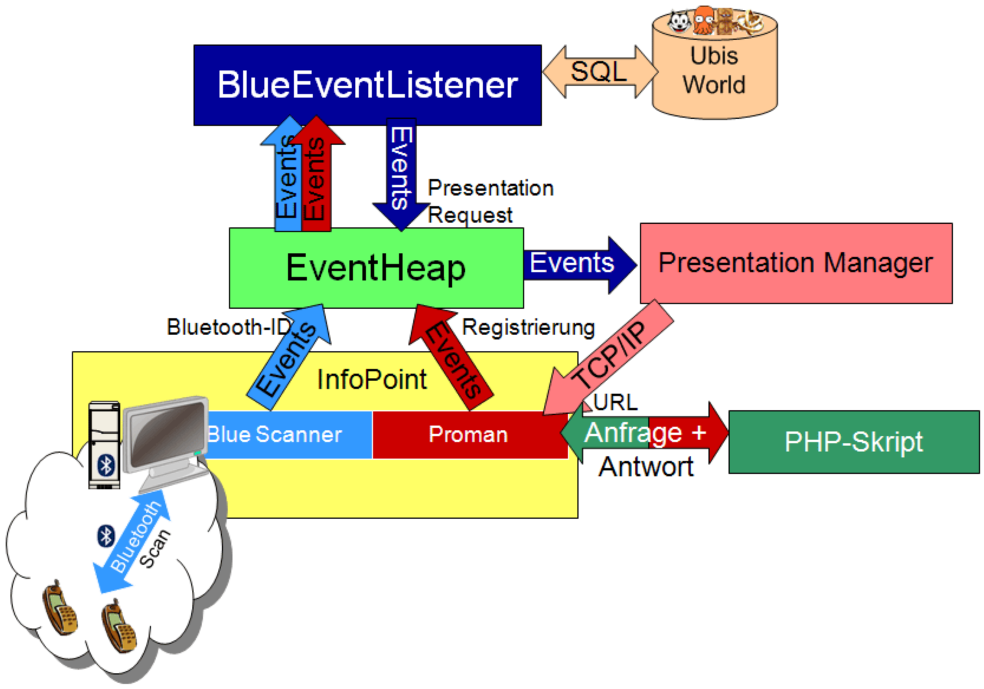


Figure 128: VISTO‘s system architecture is based on the Iplay Blue framework.

8.2.5 Automated Generation of Visual Route Descriptions

Considering the structure of buildings, one can find out that there are only a limited number of possible routes between two rooms. In the case of *VISTO*, there are even less routes to consider, since we only need route segments that begin at an infopoint display and end at another display or room in the vicinity. The route descriptions for these segments do not change unless the building is renovated in a way that walls are moved. Hence it seems reasonable to pre-compute and store all route descriptions for a building in a route catalogue. This approach saves any computation at runtime, reduces delays for the user and simplifies the system implementation. We use the *YAMAMOTO* toolkit for the modeling of the building and the PathFinder module for the route finding between all displays and destinations. Figure 129 shows all required route segments for our AI lab, where *VISTO* has been deployed. In order to access the route knowledge at runtime, we store the route catalogue in a relational database that contains one table per infopoint. Each table (see Table 13) has at least one record per destination, where 'FinalTarget' describes the destination by its UbisName and 'NextTarget' describes the next stop-over. The catalogue also includes the length between start and destination, and 'Direction' information that describes in which way the user should move when they are standing in front of the display. This direction is indicated by *VISTO* as a simple arrow. The 'MovieSource' refers to a visual route description that can be played in *VISTO's* video frame.

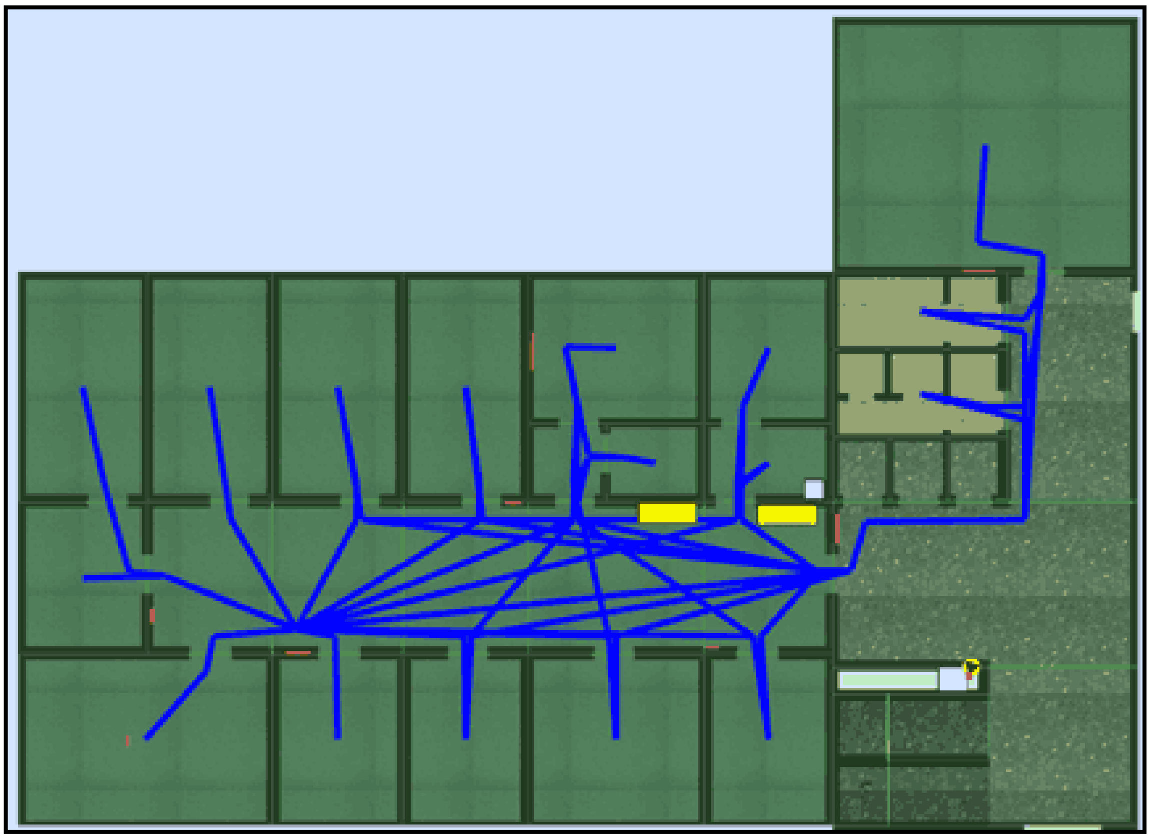


Figure 129: Overview of all route segments in our lab environment.

Table 13: The pre-computed route catalogue is represented in a relational database.

ID	FinalTarget	NextTarget	Length	MovieSource	Type	Direction
1	987275	987275	54.2636	PathCatalogueMovie0008.avi	Fast	Left
2	405012	405012	18.275	PathCatalogueMovie0029.avi	Fast	Right
3	405038	987279	13.7331	PathCatalogueMovie0009.avi	Fast	Forward
…	…	…	…	…	…	…

Finally, we use the egocentric avatar perspective of *YAMAMOTO* to visualize and export each route segment as 3-D animation. Basically, the motion of the avatar already encodes all information that is necessary for the user to follow the presented route. Still we have added meta-graphical elements to make the presentation more concise. The route itself is marked as a green line on the floor, and we inserted arrows at turning points along the route. The turning points are directly derived from the geometric course of the path, which automatically includes all decision points (Raubal, 2002), where the user needs additional information on how to continue. The arrows are placed in green color at approximately breast height into the building, pointing to the direction to turn (see Figure 130). The flat, vertical geometry of the arrows is supplemented by a shadow-like projection to the ground that helps the user estimate the position and size of the arrow. However, the described procedure of arrow insertion fails in situations with too many turning points, so that the arrows will overlap. We have also experienced that 180° turns are hardly recognizable as the route directly leaves the user's field of view. Hence we have introduced the arrow sequence as a new graphical style for curved route segments, similar to road-signs at sharp turns (see Figure 131). We also use B-spline interpolation in order to smooth the original path geometry.

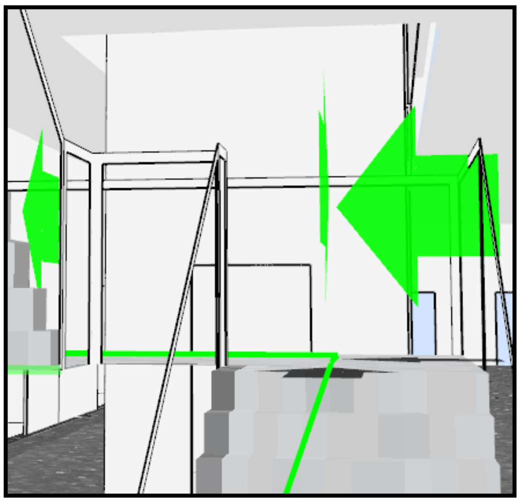


Figure 130: Arrows at turning points.

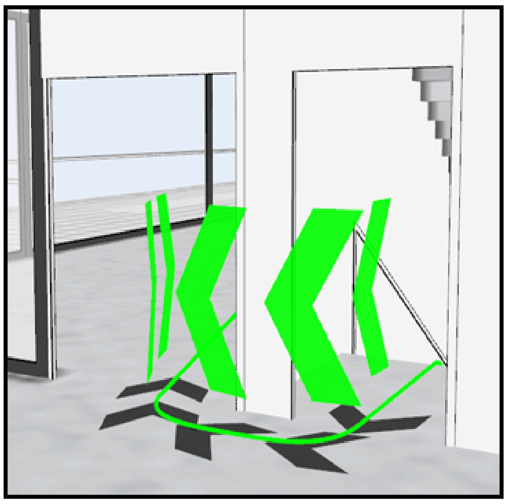


Figure 131: Arrow sequences at sharp turns.

8.2.6 Simulating and Evaluating VISTO in Dual Reality

We already discussed how we have utilized the *YAMAMOTO* toolkit to implement the visual route descriptions. We will now explain how we have used *YAMAMOTO* to simulate the *VISTO* system prior to its actual deployment in our lab.

In the first step, we have modeled several public displays in the virtual environment in order to experiment with their placement in different locations and to validate their visibility. Setting up displays in the virtual world is clearly more efficient as it avoids the need to fix mock-ups or even real display devices to the walls.

Once we have figured out suitable locations, we have tested how the proactive *VISTO* user interface would look like before we actually acquire and deploy the infopoint hardware that comprises a display and Bluetooth sensor. According to the concept of Dual Reality (see Section 6.3.5), we have connected the virtual environment model to the *Iplay Blue* infra-structure of the real environment via the event heap. We have therefore installed the *ProMan* software component together with a *VNC* server on a notebook that acts as the infopoint display. Then we have mirrored the screen of the notebook to the virtual display in *YAMAMOTO* via the *VNC* protocol, as shown in Figure 132. In order to simulate the reco-gnition of the user's mobile phone via Bluetooth, we have implemented virtual proximity sensors (see Section 7.4.1) in *YAMAMOTO* that act exactly like the real Bluetooth sensors. If the avatar enters the range of the virtual sensor, the sensor sends a recognition event with the avatar's (virtual) Bluetooth-ID to the event heap. Since the virtual and real proximity sensors send exactly the same messages, the simulation process is fully transparent to the application logic; the Iplay Blue framework reacts as if the real sensor would have discovered the user's mobile phone. The BlueEventListener sends a presentation request to the *Presen-tation Manager*, and the *ProMan* client on the notebook loads and displays the *VISTO* user interface. Through means of the *VNC* connection, the user interface also appears instant-taneously on the virtual display in *YAMAMOTO*. This simulation has helped us to develop and debug the software and user interface prior to the actual installation of the system.

Likewise, we were able to use *YAMAMOTO* for the testing of the mobile navigation device *HYBNAVI*. In the virtual environment, the *LORIOT* positioning system is simulated through the avatar, which can provide its coordinates to external applications. As the designer moves the avatar, *YAMAMOTO* sends exactly the same location events to *HYBNAVI* as the *LORIOT* positioning engine would do, see Figure 132. Hence it is possible to test the location-awareness of *HYBNAVI* prior to the actual deployment of the *LORIOT* indoor positioning system. Again, the whole simulation process is completely transparent to *HYBNAVI*.

Once a system like *VISTO* has been installed, it needs to be maintained. It is a crucial issue to monitor the client computers in a public display environment for failure, for example pop-up windows from the operating system. Our actual installation of *VISTO* only comprises three displays on a single floor, so the administration is easily manageable. However, in the case of a large installation with displays in every floor of the building, it might be cumbersome for the administration staff to check the operation of the actual displays. Furthermore, also the content has to be checked; do the labels of a door display match the room, and do the arrows of a route description point to the right direction? The virtual environment would provide an intuitive interface for the staff to monitor and validate the presentations in real time, conveniently from their office. Viewing the displays' content from an egocentric perspective in VR would probably help to identify wrong presentations (especially false wayfinding directions) easier than in a normal VNC client without spatial context.

According to the Dual Reality concept, we can also use location information in the other direction, namely tracking the real user in VR. Although the positioning engine *LORIOT* has been primarily designed to guide its user, it can also be configured to reveal and send the location information to a server. Based on this information, it is possible to visualize the user's real position in VR by positioning the avatar accordingly. One can imagine various situations where it is helpful to know about someone's where about; e.g. in industrial facilities, the tracking of engineers can help to reduce walking distances, as described in Section 6.3.7.2.

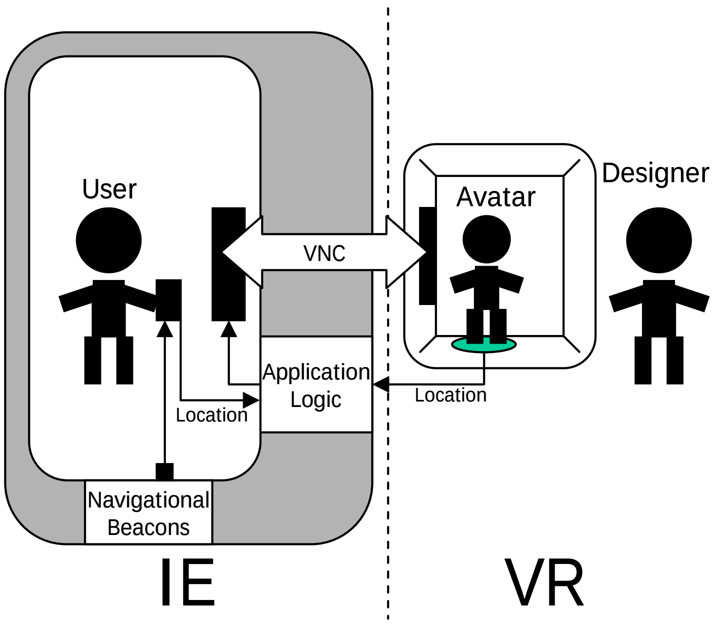


Figure 132: The instrumented environment (left) interacts with the virtual environment (right) through a VNC connection and the event heap.

8.2.7 User Study

As we have seen in the previous section, VISTO presents visual route descriptions to the user as three-dimensional animations from an egocentric viewpoint. The question arises if this format is really appropriate for the navigational task, and how it affects the wayfinding performance in comparison to other visual formats, such as maps or pictures. We assume that the egocentric perspective reduces the cognitive load of the user to mentally align the map with the surrounding reality. Furthermore, we expect fewer errors to be made in the case of routes that span multiple levels; such scenarios require the user to study multiple maps and their interrelations. However, an egocentric perspective is not particularly suitable to convey survey knowledge.

Studies on environmental learning suggest that qualities of mental representations of the environment differ (Siegel and White, 1975), depending on the learning experience (e.g. Thorndyke and Hayes-Roth, 1982; Shelton and McNamara, 2004). Route knowledge is conceived as associative memory of an ordered sequence of landmarks and directions from egocentric views, such as experienced when navigating. Route knowledge might be thus best learned if a route is presented sequentially from the egocentric perspective.

However, empirical studies on route learning in real environments do not consistently support this idea. For instance, first-time visitors of a university campus, which were presented with route instructions at an information desk, preferred maps over a sequence of view-based photographs (Devlin and Bernstein, 1995). Passively viewing a route in a virtual environment did not transfer to wayfinding success in the real building, and virtual environmental training was not superior over studying a map (Farrell et al., 2003). Indoor wayfinding behavior might be affected by a number of factors, including the complexity of the architecture and individual strategies (Hölscher et al., 2006). The present study investigates what route instruction format is most effective for route learning in a complex building.

The new computer science building at Saarland University campus was chosen as the real environment for the present interdisciplinary study conducted by the Dept. of Computer Science and Psychology (Münzer and Stahl, 2007; Münzer and Stahl, 2008). The building consists of separate functional areas with different structural features. Two partial routes were selected. The first partial route led through an open gallery system on the first floor to the library's terrace on the second floor, and the second partial route led from the library to a particular meeting room on the third floor, which comprised a corridor with side-by-side office rooms.

Experimental conditions differed with respect to the visual format of the route presentation: i) in the map condition, floor maps were shown, where the route was indicated by a line; ii) in the picture condition a sequence of pictures of decision points along the route were shown; (see Figure 133); and iii) in the animation condition a movie sequence showed the movement through the virtual building from the egocentric perspective. We have modeled the building in *YAMAMOTO* and used the different camera perspectives in order to create visual route descriptions for each condition. Whereas the animation has been created fully automatically, the perspectives for the maps and picture sequence have been chosen manually. The maps and picture sequences also needed some post-processing with image manipulation software; most notably, the arrows in the picture sequence have been

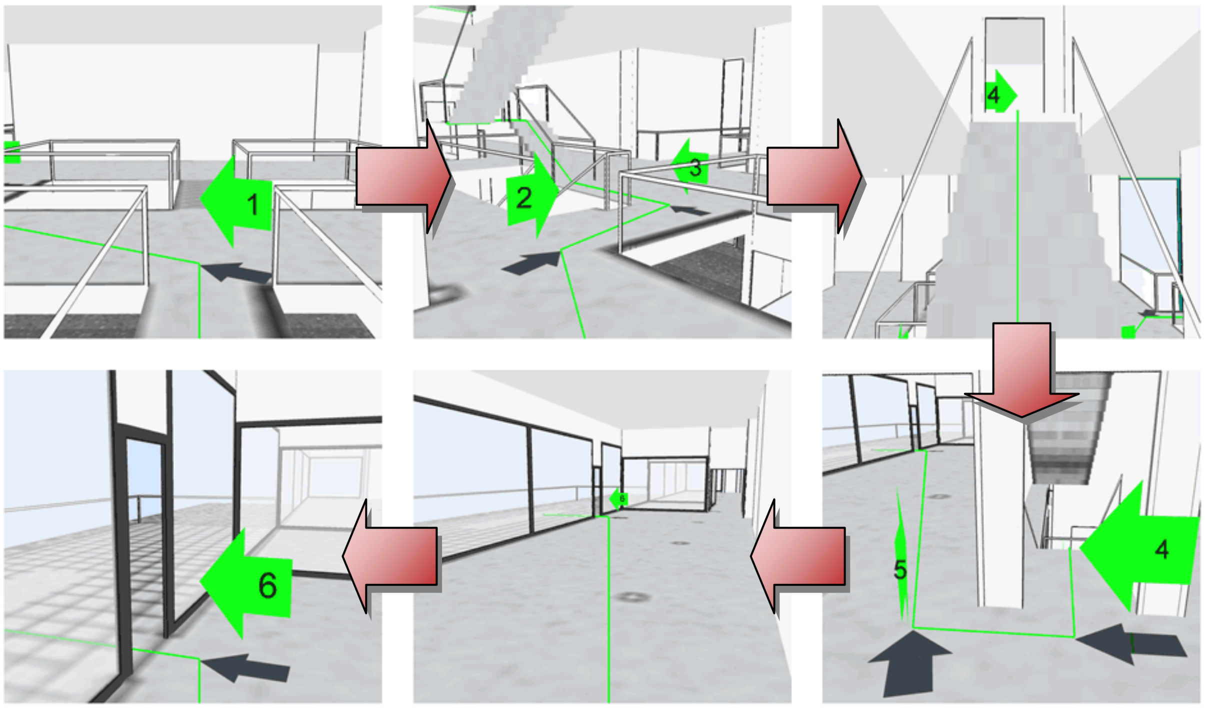


Figure 133: Part of a picture sequence; the pictures were presented sequentially to the subjects.

manually annotated with numbers so that the transition between two images becomes clearer. We have also annotated the maps and pictures with level information and partially retouched rendering artifacts.

Fourty-eight participants took part in the study (age *M* = 23.8; *SD* = 4.4). In each condition, 16 participants (8 female, 8 male) took part. Each participant was individually tested. Participants did not know the computer science building. They were paid for participation. Participants were shown the route instructions on a tablet PC at the beginning of the route. Presentations were of the same duration in each of the conditions. Participants then navigated to the destination without further assistance. In the present naturalistic scenario, they could also use information given in the environment (e.g. room numbers). The experimenter walked behind the participant and measured critical wayfinding errors (way lost) at decision points.

8.2.7.1 *Results*

Overall, 62 % of the participants made no critical wayfinding errors on the entire route, while 38% of the participants made one or more (up to five) critical wayfinding errors. The number of critical errors differed between route instruction conditions (Table 14), non-parametric Kruskal-Wallis test, qui-square (*df* = 2) = 6.835; *p* < .05. In addition, the number of participants who made critical errors in the different route instruction conditions was considered. Nine out of 16 participants who had received floor maps made critical wayfinding errors, and seven out of 16 participants who had seen pictures made critical wayfinding errors. In contrast, only two participants out of 16 who had seen animations made critical errors. These numbers differed significantly, median test, chi-square (*df* = 2) = 6.933, *p* < .05. The numbers of critical wayfinding errors made by women vs. men did not differ significantly.

Table 14. Numbers of critical wayfinding errors made by women and men in the experimental conditions.

	Maps	Pictures	Animation
Women	10	9	2
Men	9	5	0

8.2.7.2 *Discussion*

Results demonstrate a clear advantage for showing the animated virtual walk. About half of the participants who were provided either with maps (allocentric) or with sequences of pictures of decision points (egocentric) lost their ways. The advantage of the animated condition might be explained with the form of information transmission about complex turns. Turning movements on stairs, turns immediate after leaving the stairs, as well as U-turns were critical points at which errors were likely to occur. The animation transmits turning information through the movement of the virtual camera. In contrast, the same information is transmitted symbolically (by showing arrows, Figure 133) in both of the other conditions. Presumably, participants did not succeed in encoding and memorizing these symbols while passively viewing the picture sequences or the map. In contrast, when viewing the animation passively, the turning information was successfully learned due to the “analogous” transmission of that information by virtual camera movements without active encoding.

8.3 Simulation of Physical Assistance in BAALL

The Bremen Ambient Assisted Living Lab (BAALL)[69] is an automated 60 m² apartment that is equipped with adaptable furniture to support the needs of people with- and without physical impairment. The project is focused on people that depend on a wheelchair and require a lower wash basin or cooking counter than their roommates. Electronic sliding doors open automatically to let the wheelchair pass through. The wheelchair has been instrumented with sensor and actuator technology from robotics, so that it can position itself using two laser range scanners and autonomously move its passenger precisely to any desired location in the lab, avoiding obstacles along its way.

During the course of the project, one practical design problem aroused concerning the furnishing of the living space with shelves and kitchen appliances: how can the storage space be maximized without compromising the maneuverability of the wheelchair? We aimed towards a physical simulation of the wheelchair within the planned BAALL environment. According to our proposed *GOAL* method, we started our commitment to the project with the modeling of BAALL in *YAMAMOTO*. The model comprises the building structure, the sliding doors, and the currently installed as well as the planned furnishing items. We exported the geometry of the lab via RoSiML (Robot Simulator Modeling Language)(Ghazi-Zahedi et al., 2005) to SimRobot (Siems et al., 1994), a kinematic robot simulator that has been developed by the University of Bremen for research on autonomous robots.

In order to simulate the dynamic aspect of the sliding doors, we extended *YAMAMOTO* to automatically create RoSiML code that specifies a fully functional physical model of the motorized doors, comprising the tracks, a linear motor, and the framework with a door panel. It is important to say that without the semantic knowledge layer in *YAMAMOTO*, this would not have been possible. If the sliding doors were only represented as static geometric primitives without further type information, as usual in 3-D modeling tools, the RoSi-export module would not know how to construct the necessary physical elements. Table 15 shows a door's internal representation in *YAMAMOTO* as parametric box object of type *SlidingDoor* and its attributes. For a detailed description of the properties, please refer to Appendix A, Table 22. The resulting RoSiML code is listed in Table 16.

Finally, we have added an existing model of the wheelchair to the SimRobot scene that can be interactively controlled by the user. The simulation environment allows the designer to evaluate the mobility of the wheelchair in bottlenecks, such as the kitchen (see Figure 134). It is even possible to analyze collisions between the wheelchair and the sliding doors. The simulation could also be extended and used to test and evaluate the autonomous driving mode of the wheelchair in Virtual Reality. The simulation has the advantage that the virtual wheelchair is always available and can be immediately placed to any start position. In the real environment, the wheelchair also opens the sliding doors; ideally, a 2R setting would reproduce this interaction within the virtual environment. This requires the implementation of a virtual sliding door controller that mimics the behavior of the real doors.

[69] Website: http://www.baall.net

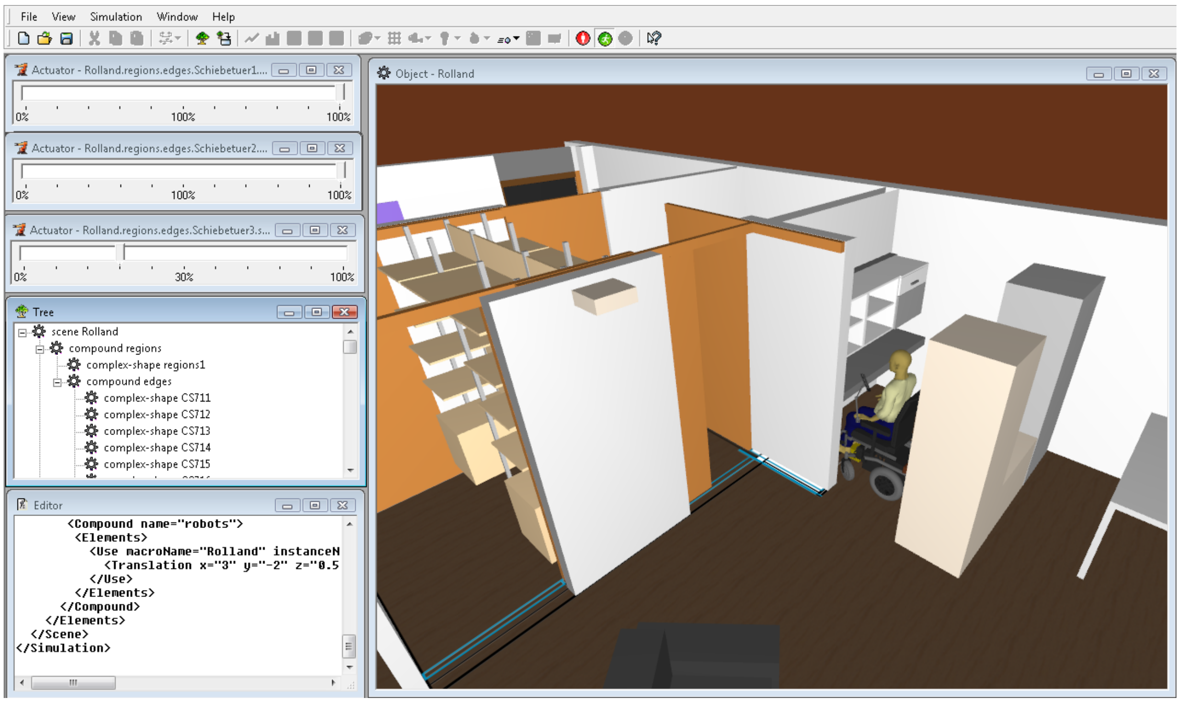
Table 15: YML code of the sliding door.

```
<Box name="Schiebetuer3" type="SlidingDoor, t=8 cm" x="990" y="1060" z="0"
orientation="0" width="210 cm" height="250 cm" depth="5 cm"
color="#CD853F" />
```

.

Table 16: The resulting RoSiML code includes physical attributes

```
<Box name="Schiebetuer3" length="2.1" width="0.05" height="0.08">
  <Translation x="-7.185458" y="6.794864" z="2.46" />
  <Rotation z="90" />
  <Appearance ref="BOXC864" />
  <PhysicalAttributes><Mass value="10"/></PhysicalAttributes>
  <Elements>
    <Slider name="sliderJoint">
      <Axis x="1.0">
         <Deflection min="0.01" max="1.05"/>
         <AngularMotor name="M1" maxVelocity="5" maxForce="100">
            <PController pControlFactor="15.0" stopBias="0.01"/>
         </AngularMotor>
      </Axis>
      <Elements>
        <Box name="panel" length="1.05" width="0.05" height="2.42">
           <Translation x="-0.525" y="0.0" z="-1.25" />
           <Appearance ref="BOXC864" />
           <PhysicalAttributes>
              <Mass value="1"/>
           </PhysicalAttributes>
        </Box>
      </Elements>
    </Slider>
  </Elements>
</Box>
```



```
        <Compound name="robots">
         <Elements>
           <Use macroName="Rolland" instanceN
             <Translation x="3" y="-2" z="0.5
           </Use>
         </Elements>
       </Compound>
     </Elements>
   </Scene>
</Simulation>
```

Figure 134: Assessing the maneuverability of the wheelchair in the kinematic simulation SimRobot.

Regarding the user interface for BAALL, the electronic sliding doors and the kitchenette can be considered as Organic User Interfaces (OUIs) (see 2.1.3.3.3), since they allow physical interaction between the user and the system. They are used for input as well as output; if the sliding doors are gently pushed sideways, this gesture is recognized as user input and communicated to the controller, which activates the motor to open or close the door. On the other hand, the assistance system can automatically open the doors to let the wheelchair pass through. Furthermore, the system can control the doors as actuators in order to adapt the configuration of the room according to the activities of the user, such as sleeping or dressing. The same is true for the kitchenette, which can be directly manipulated through the user, but also adapted by the system to fit the users' profile and tasks.

For explicit interaction, a mobile device has been attached to the wheelchair that shows a traditional, Graphical User Interface (GUI) with control elements for the environment's devices on a touch screen (see Figure 135). The menu adapts to the context of the current room, as sensed by the wheelchair's navigation system. The screenshot shows the controls for the bedroom, including virtual switches for four lights, two doors, and the bed, which can be raised at both ends. If the user enters the kitchen, the GUI changes and shows the controls to raise and lower the kitchenette.

However, controlling the environment through individual switches is not very convenient and one is likely to mistake one light with another. This problem is very similar to home entertainment systems. Whilst it is possible to control all devices with a single universal remote control, the user still has to turn on each device separately and switch the mode of the remote multiple times. A better approach is known from the Harmony[70] remote control made by Logitech (see Figure 136).

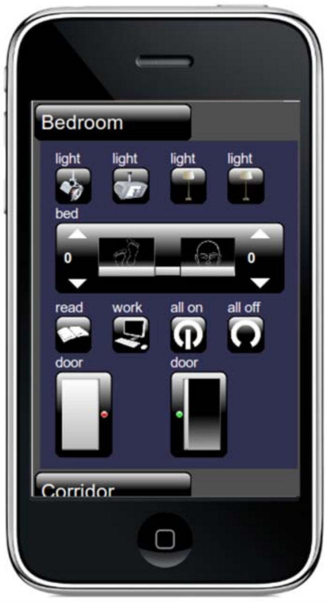


Figure 135: BAALL user interface on the iPhone. Image: DFKI Bremen.

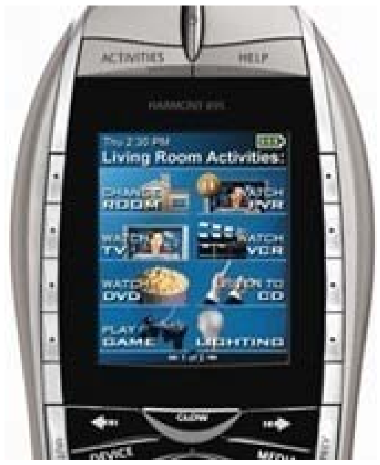


Figure 136: The activity-based Harmony universal remote control. Image: Logitech GmbH.

70 Website: http://www.logitech-harmony.com

Instead of choosing between devices, the goal-oriented user interface allows to choose from activities like “watch TV”, “listen to radio”, and “watch DVD”. Depending on the activity, the remote control automatically turns on the required devices and selects the respective input channels of the TV, receiver, and amplifier. The BAALL user interface follows this approach and, as a start, provides four macro functions that configure the bedroom for reading or working on the PC and can turn all lights on or off.

It is notable that the big achievement behind the Harmony is not the technical part, but the framework that assists the user to program the remote control according to the individual combination of components at home. The user is guided step-by-step through the setup on the PC, and the wizard provides a list of almost all components that are on the market. Similarly, the activity editor of *YAMAMOTO* could support activity-based configurations of ambient assisted living environments like BAALL by defining the relationship between activities, artifacts, and places. *YAMAMOTO* could also aid the integration of new items to the environment, such as an additional lamp, a new display, or a set of additional speakers.

In the long term, BAALL aims towards a goal-oriented user interface that is primarily based on natural language and allows its users to verbally state their needs. If the users’ intentions are known, the system can also proactively deal with necessary operations; for example automatically switch on lights and reconfigure the adaptable furniture elements. However, it is not always evident in which way the devices should react to certain situations and intentions. Hence we plan for the future to model the activities in BAALL according to the GOAL method in order to analyze them for assistance opportunities.

The joint work with the scientists in the lab also revealed the need for good debugging tools, as suggested in Section 6.3.7.3 of this thesis. For example, it is common to use programmable multi-purpose buttons in the domain of home automation. The problem is that the actual assignment between buttons and control functions is invisible to the user and developer, so engineers have to figure out the functionality of a soft button either by reading logfiles or simply by trial and error. The proposed visual inspection of the environment in a Dual Reality model would help the engineers (and users) understand such situations more quickly. Other problems were caused by syntax errors in the communication between software components, mostly typographic errors in object identifiers. The use of ontologies like *UbisWorld* could help minimize such faults. Overall, the engineers managed to solve the described problems quite well through close cooperation. However, in a real-world project outside the lab, such cooperation is not always possible, and would at least cause high travel expenses. Therefore, development strategies like the GOAL method and *YAMAMOTO* toolkit seem to be a necessity for the future of Ambient Assisted Living, similar to integrated development environments (IDE) in the software industry today.

8.4 Navigational Assistance in a Supermarket

The Innovative Retail Lab (IRL) develops assistance applications for customers in retail environments, such as supermarkets (see also Section 3.1.2.2). In cooperation with Saarland University, the IRL has early adopted the *YAMAMOTO* toolkit to model their 500 m² living lab and to plan the necessary instrumentation with sensors and interaction technology. Besides various shelves with public displays, the lab has been equipped with a steerable projector, as known from the SUPIE environment (Chapter 5), and an RFID-floor for indoor positioning.

8.4.1 Instrumenting the Supermarket and Shopping Cart

The deployment of the RFID floor has been a challenging endeavor, since several hundred passive RFID tags needed to be laid out in a dense raster in order to provide the desired positioning accuracy. It is absolutely necessary that the position of each tag is exactly known by the positioning system, so that the system can infer the location of the user from the received tag IDs. Hence the tags and their IDs have been modeled in Yamamoto, as shown in Figure 137. Since this procedure required much effort and some tags were broken later during the installation of the carpet, a more effective backup solution has been additionally installed, where only the shelves are marked with RFID tags.

The navigation system has been installed on a shopping cart, which has been instrumented with an RFID antenna and reader at the bottom and a small tablet-PC that is mounted on the cart's handle, see Figure 138. The shopping cart is a special prototype made from plastic, so that its structure does not drain electromagnetic energy of the RFID signal.

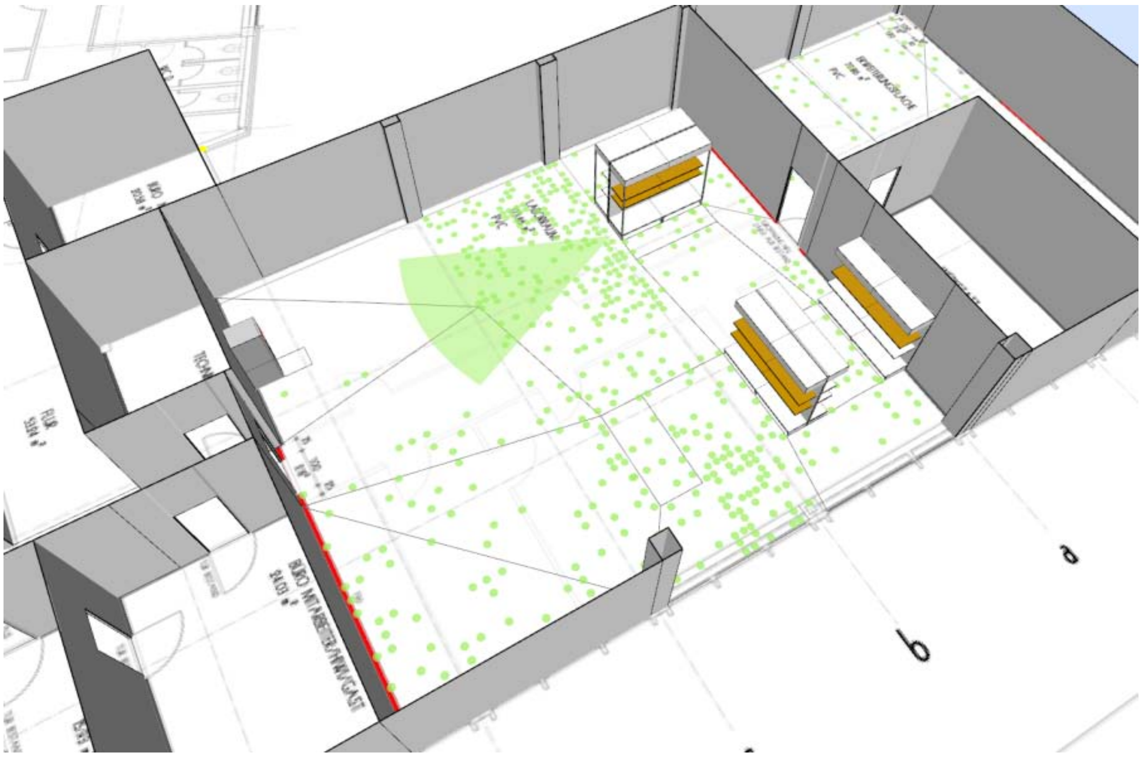


Figure 137: Yamamoto model of the IRL, the RFID tags on the floor are clearly visible.

8.4.2 Finding the Optimal Route

The requirements for the navigation application are twofold. From the customer's perspective, the system should guide the user directly to the desired product, following the shortest path. In case of a (shopping) list that contains multiple products, the shortest round-trip route along each product should be chosen, which is also known as the traveling-salesman problem in computer science. In the special case of frozen food, the system should also consider to pick up frozen food at last. From the vendor's viewpoint, marketing aspects play an important role. Guiding the customer quickly through the store reduces the chance of bargain sales and chance purchases. Since the system will be installed on behalf of the store management, it should consider advertisements and special sales. It is also desirable for the management to balance the number of customers at the sales counters and cash points in order to avoid queues.

Hoffmann (2009) has realized an in-store navigation system that meets the requirements above by utilizing the Yamamoto location model and the PathFinder algorithm that finds shortest paths between two points. Hoffmann solved the conflicting interests between the customer and vendor by a reference path that leads from the entrance along all shelves to the cash point. This path is edited by the vendor and implicitly defines a fixed order in which the products are visited. In case of multiple items, the shopping list will be sorted according to this order. Then the user will be guided from one item to the next.

8.4.3 User Interface

The interaction between the customer and the system is initiated through the user by scanning the fingerprint on a sensor at the shopping cart. The system, which is mounted to the shopping cart as shown in Figure 138, greets the customer and downloads the shopping list by wireless LAN. The navigation assistance system supports three modes of operation:

- In *passive* operation, the user walks through the store as normal. If the system recognizes that a product from the shopping list is nearby, it reminds the user of this item;
- In *active* operation, the user selects an item or destination through the touch screen. The system guides the user then directly to this product;
- In *shopping list*-based operation, the system sequentially guides the user to each item on the list. This mode reduces the chance of detours caused by forgotten items.

The display shows an abstract graphical map of the surrounding area, depending on where the user is positioned, see Figure 139. The position and orientation of the user is indicated by a green triangle. Shelves are depicted as yellow areas, and destinations are marked with a red bull's-eye and a numerical index, according to the shopping list. The suggested path is indicated as sequence of arrows. The user can also zoom into the map.

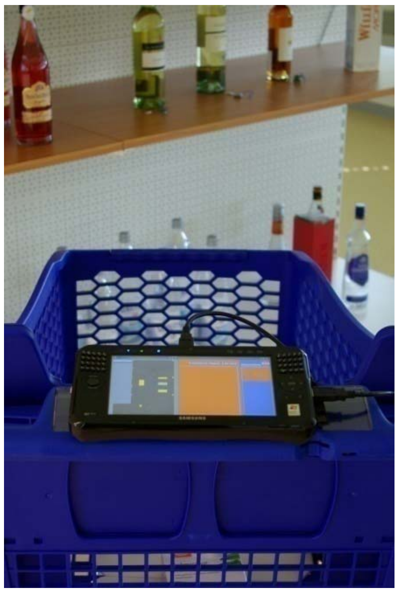


Figure 138: Cart with assistance.

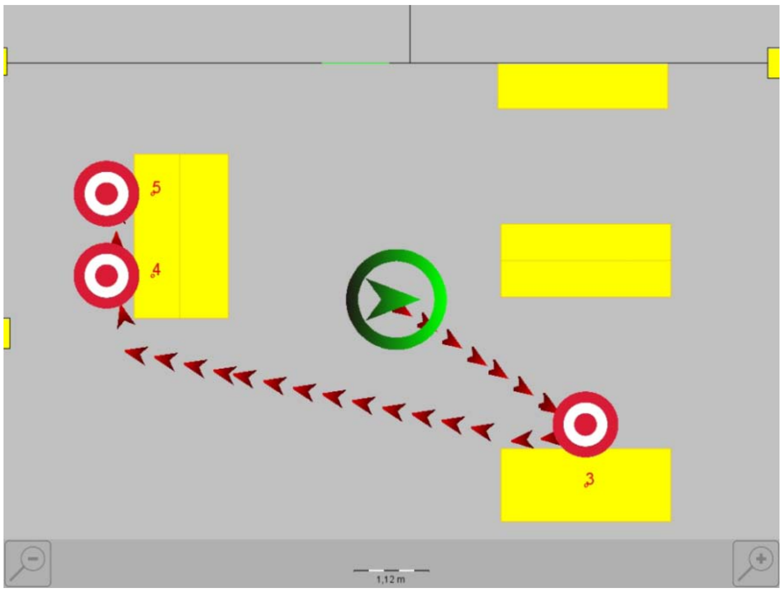


Figure 139: Suggested route according to the shopping list.

8.5 Modeling and Simulating a Zoo Environment

For the development of the *Roaring Navigator* zoo guide (Stahl, 2007) *Yamamoto* has been used to create a geo-referenced, geometric location model of the Saarbrücken Zoo environment and to test the application in VR. The *Roaring Navigator* is a location-based application running on a mobile device that implements a proactive tour guide. This project differs from the previous use cases since it is situated in an outdoor environment.

The user interface of the *Roaring Navigator* is based on a shared auditory landmark display which conveys spatial survey knowledge and navigational aid to multiple users. Our guide is situated in a zoo environment, so we use recordings of animal voices to indicate the location of the animal enclosures. Spatial audio manipulates playback of the sound clips, so that the listener can identify their distance and direction. The system also proactively presents audio clips with detailed information about each animal.

Yamamoto has been used to markup the animals' enclosures on a geo-referenced map of the zoo as circles and to associate them to a database of sound recordings. The resulting zoo model has been exported for the roaring navigator application on the PDA in Yamamoto's *YML* file format.

We will first present the concept of the user interface, which is based on an auditory landmark display. Next we explain the usage of Yamamoto for modeling, and testing in VR.

8.5.1 Motivation

Electronic tour guides are already widely used in museums. They provide access to similar information as in a guided tour, but they still allow the user to take their time and communicate with their companions. Several research prototypes have implemented location-aware user interfaces which automatically activate contents. However, since the played audio clips typically are relatively long, audio guides tend to isolate visitors and additional graphical user interfaces detract the attention from the exhibits.

Our aim is to develop an electronic tour guide which combines two functionalities: First, the guide should provide detailed information on exhibits using audio but without the negative effect of isolating the user. Second, the system should provide navigational aid and spatial survey knowledge about the environment without distracting the user from the sights and exhibits while moving. Therefore we introduce the concept of an auditory landmark display.

We use two *HP-IPAQ 5550* PocketPC PDAs which are connected by a wireless LAN ad-hoc network that covers a distance of 40 meters outdoors. For positioning, we use a Bluetooth GPS receiver in combination with a *Vectronix DRC* digital compass device. This is a very precise 3-D magnetometer that was clipped onto the back of a baseball cap. We have prepared two almost identical instances of the prototype, so that two persons can use them for a shared zoo experience, but the system architecture would also allow for more devices.

8.5.2 Auditory Landmark Display

According to John G. Neuhoff, an *auditory display* is the use of non-speech sound to present information. We introduce the new concept of an **auditory landmark display** to help the user to explore an unknown environment. It conveys information about the location of points of interest to the user, but also communicates some basic information about the places themselves. Our auditory display employs spatial audio techniques to create multiple virtual sound sources that appear to emanate from a certain direction and distance relative to the listener. Through estimating the position and orientation of the listener's head by means of a GPS receiver and magnetometer, the display aligns virtual sound sources with the true geographic position of the landmarks. The spatial audio effect is currently realized through adjusting the volume of the left and right stereo channel on the headset, as shown in Figure 142. The more distant the virtual sound sources are, the quieter they are played. Landmarks behind the user are muted to avoid confusion between the front and the rear.

Holland (2002) has presented the *Audio GPS* system, a spatial audio display that uses an abstract sound to support navigational tasks. Our work is inspired by the concept of auditory landmarks and aims to represent spatial survey knowledge. By carefully choosing sounds that one would typically associate with the landmarks, the auditory display is able to convey a rough sense of their type. The display does so through an ambient soundscape without the need for spoken language or graphics. For example, a church might be represented by a bell, a restaurant by the noise of dishes, or a roller coaster by screams of excitement. Thus, the auditory landmark display represents a minimum attention user interface with reduced cognitive load, which is useful in situations where the user should be focused on their surroundings instead of the mobile device. Besides exploration, the landmark display also supports navigational tasks by the frequent repetition of a target landmark.

8.5.2.1 *Exploration Mode*

In the exploration mode, the auditory display continually plays the three closest landmark sounds in a randomized order. Figure 140 shows a map of the zoo and auditory landmarks *restaurant*, *flamingo*, and *jaguar*, which are nearby the user. Each landmark is surrounded by a circular activation zone which is extended by an outer proximity zone (see Figure 143). As the user enters the proximity zone for the first time, the system gives an explanatory comment in pre-recorded speech. For example, in the case of the gibbons, the voice says "*you are close to the gibbons*". Upon entering the activation zone, the system automatically starts the playback of an audio clip which describes the animal in detail.

8.5.2.2 *Navigation Mode*

Via the menus of a graphical user interface the user can switch to the navigation mode by selecting a certain animal from a list. Now the destination landmark is repeatedly played, until the user reaches the activation zone. In Figure 141, the yak has been selected which is located in the upper right of the map. No route planning is done by the system, based on the assumption that in a pedestrian environment with a dense path network no explicit directions are necessary.

We have conducted an initial user study of paired visitors in the zoo to evaluate the usability of the system with positive results (Stahl, 2007). The participants reported that the system is easy to use and has a stimulating influence on the communication between the visitors. As a further result, the study indicates that 'lightweight' navigational aid can be sufficient for wayfinding tasks in certain environments, which provides only the linear distance and direction of the destination. Figure 141 shows a map with three typical paths (black, white, grey) that have been taken by the subjects during the study.

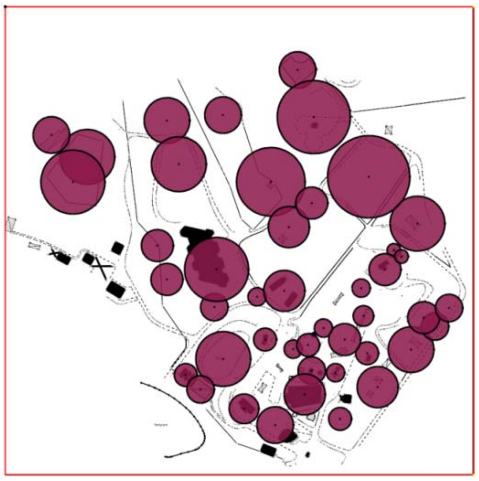


Figure 140: Location model of the zoo.

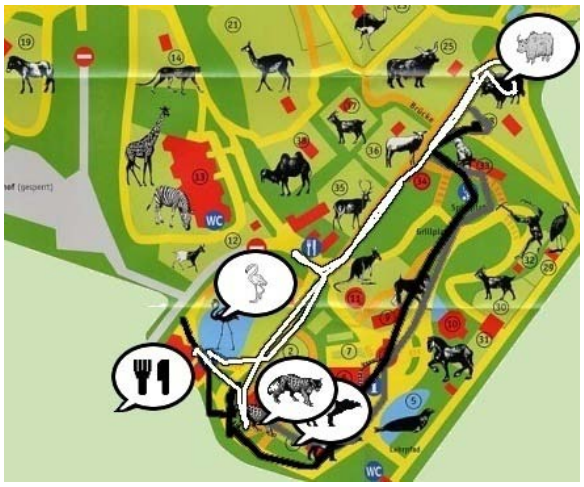


Figure 141: Results of the wayfinding task.

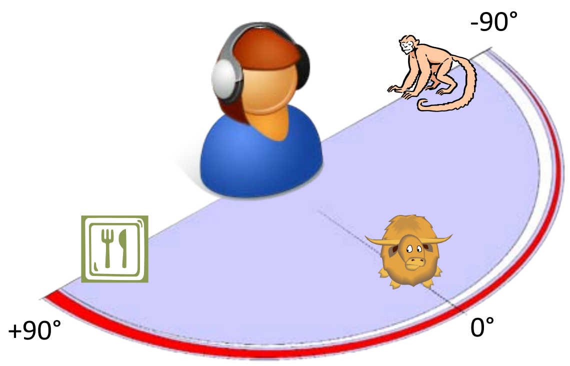


Figure 142: The spatial audio effect is realized by adapting the headphone's stereo balance.

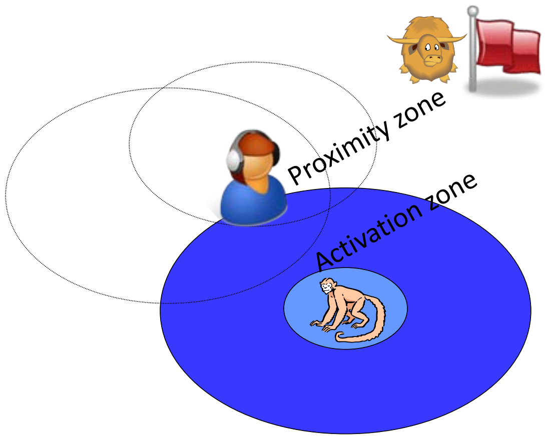


Figure 143: Activation and proximity zones.

8.5.3 Simulating and Testing in VR

Testing the application on location turned out to be time consuming; before each test we had to drive from the university to the zoo, set up the mobile equipment comprising two PDAs with external GPS receivers and magnetic compass units, and we had no opportunity to recharge the batteries on location. The intended routes for the user study covered a large spatial area with considerable walking distances (about 15 minutes), and we were depending of good weather conditions, since the PDAs are not water resistant.

In order to minimize the testing on location, we have extended Yamamoto to simulate a real GPS receiver unit, based on the avatar position within the virtual zoo model. Yamamoto converts the model coordinates of the avatar into geographic coordinates according to the *WGS84* model and encodes these coordinates into *NMEA 0183* sentences (see 7.4.2). Yamamoto outputs the sentences on the serial port so that the *Roaring Navigator* application on the PDA can directly read them, as if they were coming from a real GPS receiver unit. This testing method is fully transparent to the mobile application; it allows the developer to evaluate the behavior of any mobile application without changes of the original code. In previous projects, such as the *BMW Personal Navigator* (see 5.2.1), additional debug code has been required to test the behavior of the mobile application in the lab by manually changing the position of the user. However, such testing methods do not prove that the program will work with actual GPS data on location. Furthermore, experience from the earlier projects has shown that debug code might be unintentionally left and cause abnormal behavior in normal operation. Testing with *YAMAMOTO* has been safe and considerably reduced the need for debugging and testing of the location-awareness in the field.

8.6 Summary and Discussion

In this Chapter we have presented five use cases where the GOAL method and *YAMAMOTO* toolkit have been applied in the development process, namely the SmartKitchen project, the *VISTO* indoor navigation system, the Bremen Ambient Assisted Living Lab (BAALL), the IRL, and the *Roaring Navigator*, a mobile tour guide for the Zoo Saarbrücken. Table 17 gives an overview on how the *GOAL* method and the *YAMAMOTO* toolkit have contributed to different aspects of system development.

The first use case describes how we have applied *YAMAMOTO* to geometrically model the *SmartKitchen* environment, including the building structure, kitchen appliances, and shelves with boards. All geometric elements have also been symbolically modeled in the UbisWorld ontology, so that for example each location in a shelf has got a unique identifier that can be used to express the location of artifacts or ingredients. Regarding the modeling of activities, we have used the activity editor to represent the cooking activities and user assistance given by the *SmartKitchen's Semantic Cookbook* application. The kitchen's instrumentation for human-environment interaction has been represented in detail, including the display, some RFID readers, and the control cube, a tangible user interface. During the implementation phase, *YAMAMOTO* has been used to create movie sequences for micro-navigational aid.

In the second Section, we described the development of the activity-adaptive indoor navigation system *VISTO* that provides visual route descriptions to pedestrians. Similar to the *SmartKitchen*, we have used *YAMAMOTO* to model the building structure, landmark objects, and public displays that serve as user interface. In this use case, we have highlighted *YAMAMOTO*'s ability to represent multiple levels, which was a prerequisite for the automated generation of route directions. The geometric location model has been extended to a hybrid model, since symbolic names for the rooms were required to build a database with route information. Regarding the modeling of activities, we have modeled a list of *spatial purposes* that can be associated with places in the *UbisWorld.* We also described how we have used *YAMAMOTO* for the implementation of the indoor positioning system. Finally we explained how we have simulated the user-adaptive interface in a Dual Reality setting.

The BAALL use case describes how we have applied *YAMAMOTO* together with the DFKI's robot simulator *SimRobot* in order to interactively evaluate furniture configurations with respect to the maneuverability of driving assistants. Therefore, we created a detailed model of the living lab, including its shelves, kitchen appliances, and sliding doors. We implemented an export functionality for *SimRobot* that gains advantage of *YAMAMOTO*'s semantic meta data to create a fully functional physical model of the doors. In the remainder of this use case, we have also discussed how the activity editor could be applied to support the design of an activity-based remote control.

In the following section, we reported on how the Innovative Retail Lab (IRL) has adopted *YAMAMOTO* and the PathFinder algorithm to develop an in-store navigation system for supermarket customers. Initially, they aimed for an RFID-enhanced floor to support the positioning of the shopping cart, but later decided for beacons at the shelves. The implementation of the navigation system also had to reflect the interests of the supermarket management; whereas customers probably prefer shortest paths, the management's intention is to route shoppers along advertisement areas. We have presented the IRL's solution that is based on a round-trip path.

Finally, we presented the Roaring Navigator use case, where we describe the implementation of a zoo guide that indicates nearby animals through a spatial audio display. The novel aspect is that we use sound samples of the animal's voices instead of speech, which results in an ambient, minimal attention interface that doesn't distract visitors from the actual sights around them and allows communication with others. We have used the *YAMAMOTO* toolkit for the modeling of the zoo, and the implementation benefits from the coordinate transformation matrix to map the user's GPS position to the zoo model. During the evaluation phase, the GPS emulation feature helped to debug the Roaring Navigator in the lab based on a simulated position signal, even without modifications on the code.

Table 17: Overview of the five presented use-cases

		SmartKitchen	**VISTO**	**BAALL**	**IRL**	**Roaring Navi.**
Geometry						
	Building	Kitchen room, interaction zones	Buildings E11 and E13, multiple levels and stairs	Sliding doors	Supermarket floor plan	Outdoor environment, Animal enclosures
	Furnishing	Shelves	Tables, Cabinets	Shelves, Tables, Bed, Sofa, Kitchen	Shelves	-
	Artifacts	Kitchen appliances	-	Kitchen Appliances, PC	-	-
Ontology		Full domain	Displays, locations	-	Products, Shelves, spatial regions	Database of acoustic landmarks
Activity		Bruschetta with and without assistance	Ubidoo task planner	-	-	-
Human-Environment Interaction		Display, RFID antenna and reader, Control Cube	Displays, RFID Beacons, IR beacons, Bluetooth Dongles	-	RFID reader	Location-based services: activation zones
Implementation		Micronavigation	Indoor positioning, automated generation of 3-D route descriptions	-	Location model for route finding, Pathfinder algorithm	Coordinate conversion from GPS to model
Simulation & Evaluation		-	Dual Reality: Avatar motion, Bluetooth sensor, VNC displays content for user study	Physical simulation of robotic wheel-chair, sliding doors	-	GPS and compass emulation

9 Conclusion

The main contribution of this thesis is the GOAL design method that aids the design process of user assistance systems in intelligent environments through the YAMAMOTO toolkit, which integrates a geometric environment model with a situational semantic activity model that has been derived from Activity Theory. The activity model represents the users' needs in a structured form through the use of ontological concepts which link activities with their spatial context in the environment model.

On a practical level, the *YAMAMOTO* toolkit allows the designer to create a 3-D model of the environment from a set of parametric objects that are used to represent the building's structure, furnishing, and instrumentation with relatively little effort compared to traditional editors for 3-D graphics. The integrated Activity Editor allows the designer to define and browse through activities and to visualize their context in the environment. Interaction design is supported by *YAMAMOTO* through an ontology of available sensors and actuators and their placement in the virtual model, including the assessment of their range of sensitivity and effect, e.g. the visibility of a display.

We have also outlined Dual Reality (2R) applications of *YAMAMOTO,* such as remote debugging, monitoring, and maintenance of instrumented environments. 2R takes advantage of the symmetry between the virtual model and the real environment by coupling them by means of sensor/actuator networks, so that they can mutually reflect and influence each other.

In the following, we will highlight some of the results that were achieved throughout this work in more detail and report on our collaborations with partners from industry and academia. Finally, we conclude this chapter with a discussion of opportunities for further research based on the findings of this work.

9.1 Scientific contributions

We have developed the graphical map modeling toolkit *YAMAMOTO*, which has matured throughout this work from a prototype to an application that is used by several research projects for the modeling of built environments. To our best knowledge, it is the first system that consequently applies semantic annotation in order to connect the geometric entities of the environment model with symbolic instances that are represented by an ontology, thereby forming a hybrid location model. From the use of semantic knowledge we gain the following advantages that are worthwhile to highlight:

- **Efficient 3-D visualization from a 2.5-dimensional draft.** Buildings and rooms are modeled in *YAMAMOTO* as spatial regions, which are represented by their outline through sets of vertices and edges. Multiple levels can be modeled as horizontal planes that are arranged along the z-axis in 3-D space. Although the spatial regions are basically represented as flat objects, they can be visualized from an egocentric avatar perspective in full 3-D through parametric objects. These objects automatically generate the necessary geometry for walls, doors, and windows for each edge from a set of symbolic properties and measures. Furthermore, a set of generic furnishing objects exist to easily add tables, shelves, appliances, and displays to the model.

- **Seamless route-finding in indoor and outdoor environments without the explicit modeling of path networks.** The *YAMAMOTO* toolkit includes the PathFinder component that has been implemented to find shortest paths in multi-level building models. The hierarchic modeling approach also allows for room-to-room routing between different buildings that are connected by footpaths, e.g. on a campus. The symbolic annotation of regions by their accessibility (pedestrians, cars, wheelchairs) and whether edges are passable (and in which direction) allows the algorithm to perform an A* search directly on the spatial regions. The PathFinder algorithm automatically creates the required path network segments on the fly as the search commences. This leads to natural routes that cross regions as necessary, similar to the actual wayfinding behavior of pedestrians.
- **A hybrid location model that seamlessly spans spatial granularities from continents to shelf boards.** Spatial regions and storage spaces of the geometric environment model can be linked to their symbolic counterparts in the UbisWorld in order to represent spatial relations, such as *part-of* and *inclusion*. In conjunction with the geographic UbisEarth ontology, the symbolic location identifiers are particularly useful to denote the location of objects within the world. For example, it is possible to model a library with shelves, and to relate the shelves to geographic entities, like building, street, and city. The symbolic identifier of the shelves can now be used to describe the location of books within the UbisEarth tree, similar to a postal address. Furthermore, additional knowledge about objects and locations can be represented in UbisWorld, such as international labels, and facts.
- **Dual Reality-Ready.** The hybrid modeling approach allows to link elements of the geometric model to their real counterparts via symbolic identifiers. Assuming that these IDs are known, it is possible to query the instrumented environment for the state of its sensors and to control the actuators. However, the environment has to provide the necessary interfaces and *YAMAMOTO* needs to implement appropriate driver modules for the virtual sensors and actuators.

We have further introduced *GOAL*, a structured, user-centered design method for user assistance in instrument environments, particularly in the ambient assisted living (AAL) domain. The method is based on three pillars: i) a 3-D model of the environment; ii) a situational semantic activity model that represents the users' needs; and iii) the *UbisWorld* ontology which is used to relate the activities to their context. The *YAMAMOTO* toolkit provides an integrated user interface for the hybrid modeling of geometry and activity, which contributes novel solutions to the following design aspects:

- **Formalizing users' needs into a situational semantic activity model.** It is of critical importance for the success of assistive technology for everyday life scenarios to assess and capture the actual needs of the target group, so that system designers and developers can put themselves in the position of the user. Especially in the Ambient Assisted Living domain it is difficult for the user interface designer to anticipate the limited (sensory, motor, and cognitive) abilities of elderly users. Typical elicitation techniques include ethnographic studies and interviews with target users. These methods result in individual and subjective statements that need to be formalized into a structured design specification. Our toolkit supports the transformation of informal, verbose statements into a tabular representation of activities and actions through the Activity Editor component.

- **Interaction design decisions for sensors and actuators.** The integration of the activity model with the geometric environment model allows the designer to browse through activities and to highlight their context in the environment, considering spatial regions, artifacts, and the instrumentation with sensors and actuators. The *YAMAMOTO* toolkit supports design decisions by an ontology of available device classes and products that provides an overview about the available choice of sensors and actuators to the designer.
- **Assessing the range of sensors and actuators.** The *YAMAMOTO* editor graphically supports the instrumentation of the environment by placing sensors and actuators in the virtual environment. This step includes the assessment of their range, for example the visibility of a display, a camera's field of view, and the distance between RFID readers and tags.
- **Interacting with instrumented environments in Dual Reality.** On a conceptual level, we have introduced four scenarios how the emergent property of 2R environments can be utilized in domotic and industrial settings for the remote debugging, monitoring, and controlling of instrumented environments. Practically, we have realized the connection between our lab and its virtual model in *YAMAMOTO*. In this setting, we have mirrored the content of real public displays into the virtual world. As a real user approaches the display, a Bluetooth-based proximity sensor recognizes the user's profile and the display adapts its presentation accordingly. This effect can be observed in the virtual world. Vice versa, the real display also responds to the proximity of an avatar in front of the virtual display.

9.2 Contributions to the Industry and Research Community

In Chapter 8 we presented some of the results that we have achieved with the *YAMAMOTO* toolkit in projects at Saarland University and DFKI. The toolkit is also used in several external projects for the spatial modeling of built environments and instrumented environments. *YAMAMOTO* has been licensed through collaboration contracts to the following partners:

- BMW Forschung und Entwicklung GmbH, Munich
- Prof. Dr. Ohlbach, LMU Munich
- XevIT GmbH, Karlsruhe
- Agilion GmbH, Chemnitz

Furthermore, *YAMAMOTO* is currently used by T. Kuflik at the University of Haifa, Israel, to model the Hecht Museum, and by Prof. I. Kobayashi at Ochanimizu University to model the Ocha House (3.1.1.10) in Tokyo, Japan.

YAMAMOTO and the visual route instructions (VISTO) have been exhibited to the public as solution for pedestrian navigation in indoor environments at the CeBit Fair (Hannover, March 2007). This presentation has been covered by newspapers (Frankfurter Allgemeine Zeitung/FAZ.NET), television (RTL Nachtjournal), and live on radio (Deutschlandfunk). We were also invited to discuss the applicability of route instructions on public displays with the management of Frankfurt Airport (FraPort AG).

In conjunction with the 8th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI'06, 12 to 15 September 2006, Espoo, Finland), we (Stahl et al., 2006c) have organized the MODIE (Modeling and Designing Intelligent Environments) workshop in order to explore the research questions of this thesis with other researchers.

9.3 Opportunities for further research

For the near future we are planning to continue our work in the field of AAL at BAALL in Bremen. Furthermore, presentations have sparked interest in an industrial context at the SmartFactory in Kaiserslautern, where we aim to implement navigational aid for industrial workers. The SFB/TR 8 Spatial Cognition[71] plans to apply *YAMAMOTO* in the education of architecture students. Some further research is necessary to achieve the following goals:

- **Technical realization of a domotic 2R scenario in BAALL**. Based on the X-Gate API that grants access to BAALL's home automation systems, such as a KNX bus, we will enhance *YAMAMOTO's* user interface with the necessary control elements so that the virtual environment can be used to manipulate the real actuators. These actuators include the lights, window shades, sliding doors, and the adaptive kitchenette and bed.
- **Navigational assistance in industrial plants.** The SmartFactory is interested to adopt the developed indoor navigation solutions for factory workers, so that workers are able to quickly locate defective parts of a plant. Furthermore, a real-time visualization of the plant and the workers would be of great help to the management, so that they can assign maintenance jobs based on actual walking distances within the building.
- **Extending the capabilities of parametric objects**. We have currently implemented only a minimal set of objects to represent the building elements with a fairly simple geometry. However, different communicative goals require different levels of detail. In order to plan the layout and instrumentation of an environment, an abstract visual style can emphasize that the model shows just a draft. On the other hand, a final design should include all relevant details. Although these goals seem like a contradiction, they can be achieved by adding the desired level of detail as new parameter. This also comprises the modeling of walls, which are currently either modeled on an abstract level (as single edges) or as solid regions (four edges). It would be more flexible if the thickness of walls could be parameterized, so that both flat and solid representations could automatically be derived.
- **Seamless navigation between outdoor and indoor environments**. Currently, we have focused our videos for spatial orientation on indoor environments. However, pedestrians typically require route directions between different buildings that span indoor and outdoor environments. Further user studies are necessary to evaluate different designs concerning the visualization of outdoor environments and the

[71] http://www.sfbtr8.spatial-cognition.de/

transition between streets and buildings. Another question is how to deal with long passages in video animations; and how they should be abbreviated.

- **Investigate ways to convey survey knowledge to pedestrians with 3-D models.** We will continue our interdisciplinary work with the psychology department at Saarland University to find out how 3-D models can be utilized in order to convey survey knowledge to users. These experiments include studies of interaction techniques, such as panning and zooming, to explore multi-level building models in virtual reality.
- **Apply the GOAL method in real-life design projects.** The GOAL method has been used so far only for the instrumentation of the DFKI kitchen. The next steps would be to create a comprehensive activity model for everyday tasks in the BAALL environment, and to apply this model for the user interface design. Finally, it would be interesting to apply the GOAL method in other AAL projects to gain new insights.

Bibliography

Aarts, E. and Marzano, S. (Eds) (2003). The New Everyday: Visions on Ambient Intelligence. Rotterdam, The Netherlands: 010 Publishers.

Aarts, E. and Encarnação, J. L. (2006). True Visions. The Emergence of Ambient Intelligence. Berlin, Heidelberg: Springer.

Addlesee, M., Curwen, R., Hodges, S., Newman, J., Steggles, P., Ward, A. and Hopper, A. (2001). Implementing a Sentient Computing System, *Computer*, Volume 34, No. 8 (August 2001), pages 50-56. Washington, DC: IEEE Computer Society. ISSN:0018-9162.

Aipperspach, R., Hooker, B. and Woodruff, A. (2008). The Heterogeneous Home. In *Proceedings of the 10th international Conference on Ubiquitous Computing* (UbiComp '08, Seoul, Korea, September 21-24), Volume 344, pages 222-231. New York, NY: ACM.

Ajikata, S. and Kobayashi, I. (2009). A Study on TODO Task Management at Home, *International Conference on Soft Computing and Pattern Recognition* (SoCPaR2009, Malacca, Malaysia, December 4-7) (to appear)

Alexandersson, J., Zimmermann, G. and Bund, J. (2009). User Interfaces for AAL: How Can I Satisfy All Users? In *Proceedings of Ambient Assisted Living - AAL. 2. Deutscher Kongress mit Ausstellung/Technologien - Anwendungen - Management*, pages 5-9. Berlin, Germany.

Allen, J.F. and Ferguson, G. Actions and Events in Interval Temporal Logic, *Journal of Logic and Computation*, Volume 4, No. 5 (October 1994), pages 531-579. Oxford: Oxford University.

Armac, I. and Retkowitz, D. (2007). Simulation of Smart Environments. In *Proceedings of the IEEE International Conference on Pervasive Services 2007* (ICPS 2007, Istanbul, Turkey), pages 257–266. Washington, DC: IEEE Computer Society.

Arthur, P. and Passini, R. (1992). Wayfinding: People, Signs and Architecture. New York, NY: McGraw-Hill Education. ISBN: 978-0075510161.

Baber, C., Haniff, D. J. and Woolley, S. I. (1999). Contrasting Paradigms for the Development of Wearable Computers. *IBM Systems Journal*, Volume 38(4), pages 551-565. IBM Corp.

Bahl, P. and Padmanabhan, V. (2000). RADAR: An In-Building RF based User Location and Tracking System. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies* (IEEE INFOCOM, Tel-Aviv, Israel, March 26-30), Volume 2, pages 775-784. Washington, DC: IEEE Computer Society.

Bardram, J. E. and Christensen, H. B. (2004). Open Issues in Activity-Based and Task-Level Computing. In *Proceedings of the First International Workshop on Computer Support for Human Tasks and Activities*, held in conjunction with Pervasive 2004, pages 56-62. University of Aarhus, Aarhus, Denmark.

Bardram, J. E. (2005). Activity-Based Computing: Support for Mobility and Collaboration in Ubiquitous Computing Environments. *Personal Ubiquitous Computing*, Volume 9 (2005), pages 312-322. Berlin, Heidelberg: Springer.

Baus, J. (2003). Ressourcenadaptierende Hybride Personennavigation. DISKI 268. Berlin: Akademische Verlagsgesellschaft Aka GmbH.

Baus, J., Krüger, A. and Stahl, C. (2005). Resource-Adaptive Personal Navigation. In: O. Stock and M. Zancanaro (Eds.), *Multimodal Intelligent Information Presentation*, pages 71-93. Berlin, Heidelberg: Springer.

Baus, J., Krüger, A. and Wahlster, W. (2002). A Resource-Adaptive Mobile Navigation System. In *Proceedings of the 7th International Conference on Intelligent User Interfaces* (IUI '02, San Francisco, California, USA), January 13-16, pages 15-22. New York, NY: ACM.

Becker, C. and Dürr, F. (2005). On Location Models for Ubiquitous Computing. *Personal Ubiquitous Computing*, Volume 9, No. 1 (January 2005), pages 20-31. Berlin, Heidelberg: Springer.

Berendt, B., Barkowsky, T., Freksa, C. and Kelter, S. (1998). Spatial Representation with Aspect Maps. In: C. Freksa, C. Habel and K. Wender (Eds.), *Spatial Cognition - An Interdisciplinary Approach to Representing and Processing Spatial Knowledge*, pages 313-336. Berlin, Heidelberg: Springer.

Berners-Lee, T. (2000). RDF and the Semantic Web, Keynote speech at XML 2000 (Washington DC, December 6); for the “layer cake” see also www.w3.org/2002/Talks/04-sweb/slide12-0.html

Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. *Computer*, Volume 21, Issue 5 (May 1988), pages 61-72. Washington, DC: IEEE Computer Society

Brandherm, B. and Schmitz, M. (2004). Presentation of a Modular Framework for Interpretation of Sensor Data With Dynamic Bayesian Networks on Mobile Devices. In *LLWA 2004, Lehren Lernen Wissen Adaptivität*, Berlin, Germany, 2004.

Brandherm, B. and Schwartz, T. (2005). Geo Referenced Dynamic Bayesian Networks for User Positioning on Mobile Systems. In: T. Strang and C. Linnhoff-Popien (Eds.), *First International Workshop on Location- and Context-Awareness* (LoCA 2005, Munich, Germany), LNCS 3479, pages 223-234. Berlin, Heidelberg: Springer.

Brandherm, B., Ullrich, S. and Prendinger, H. (2008). Simulation Framework in Second Life with Evaluation Functionality for Sensor-Based Systems. In *Proceedings of the 2nd Workshop on Ubiquitous Systems Evaluation* (USE 2008, Seoul, South Korea). CEUR Workshop Proceedings. ISSN 1613-0073.

Buiza, C. (2008). Development of Virtual Personas to Describe Alzheimer Disease Type Users in i2home. Position paper, presented at *Capturing Ambient Assisted Living Needs Workshop* at Ami2008 Conference on Ambient Intelligence, Nürnberg, Germany.

Butz, A., Baus, J., Krüger, A. and Lohse, M. (2001). A Hybrid Indoor Navigation System. In *Proceedings of the 6th International Conference on Intelligent User Interfaces* (IUI’01, Santa Fe, New Mexico, USA, January 14-17), pages 25-32. New York, NY: ACM.

Butz, A., Schneider, M. and Spassova, M. (2004). SearchLight - A Lightweight Search Function for Pervasive Environments. In: A. Ferscha and F. Mattern (Eds.), *Pervasive Computing* (Pervasive 2004), pages 351-356. Berlin, Heidelberg: Springer.

Carey, R. and Bell, G. (1997).The Annotated VRML97 Reference Manual. Essex, UK: Addison-Wesley Longman Ltd. Website: http://www.cs.vu.nl/~eliens/documents/vrml/reference/BOOK.HTM

Carroll, J. M. (1990). The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill (Technical Communication Series). Cambridge, MA: MIT.

Carroll, J. M. (2000). Making Use: Scenario-Based Design of Human-Computer Interactions. Cambridge, MA: MIT, 2000.

Cheverst, K., Davies, N., Mitchell, K. and Friday, A. (2000). Experiences of Developing and Deploying a Context-Aware Tourist Guide: the GUIDE Project. In *Proceedings of the 6th Annual international Conference on Mobile Computing and Networking* (MobiCom'00, Boston, Massachusetts, United States), August 6-11, pages 20-31. New York, NY: ACM.

Cheverst, K., Fitton, D. and Dix, A. (2003). Exploring the Evolution of Office Door Displays. In: K. O’Hara, M. Perry et al. (Eds.), *Public and Situated Displays: Social and Interactional Aspects of Shared Display Technologies*. Chapter 6, pages 141-169. Cologne: Kluwer.

Christensen, H. B. and Bardram, J. E. (2002). Supporting Human Activities – Exploring Activity-Centered Computing. In *Proceedings of 4th International Conference on Ubiquitous Computing* (UbiComp 2002, Göteborg, Sweden), LNCS 2498, pages 107-116. Berlin, Heidelberg: Springer.

Churchill, E., Nelson, L., Denoue, L., Helfman, J. and Murphy, P. (2004). Sharing Multimedia Content with Interactive Displays: A Case Study. In *Proceedings of the 5th Conference on Designing interactive Systems: Processes, Practices, Methods, and Techniques* (DIS '04, Cambridge, MA, USA), August 1- 4, pages 7-16. New York, NY: ACM.

Coad, P. and Yourdon, E. (1991). Object-Oriented Analysis, 2nd Edition. Englewood Cliffs, NJ: Yourdon, Prentice Hall.

Coen, M. (1998a). Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments. Spring Symposium Series Technical Reports, TR-SS-98-02. AAAI.

Coen, M. (1998b). Design Principles for Intelligent Environments. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (AAAI'98, Madison, WI, USA), pages 547-554. AAAI.

Cooper, A. (1999). The Inmates are Running the Asylum. New York, NY: Macmillan.

De Ruyter, B. (2003). 365 Days' Ambient Intelligence Research in HomeLab. Eindhoven, The Netherlands: Philips Research Public Relations.

De Ruyter, B., Aarts, E., Markopoulos, P. and Ijsselsteijn, W. (2005). Ambient Intelligence Research in HomeLab: Engineering the User Experience. In: Weber, W., Rabaey, J. and E., Aarts (Eds.), *Ambient Intelligence*, pages 49-62. Berlin, Heidelberg: Springer.

De Ruyter, B. and Pelgrim, E. (2007). Ambient Assisted-Living Research in CareLab. ACM *interactions* Journal, July/August 2007, pages 30-33. New York, NY: ACM.

De Ruyter, B., van Loenen, E. and Teeven, V. (2007b). User Centered Research in Experience Lab. In: B. Schiele et al. (Eds), *Proceedings of Ambient Intelligence 2007*, pages 305-313. Berlin, Heidelberg: Springer.

Devlin, A. S. and Bernstein, G. (1995). Interactive Wayfinding: Use of Cues by Men and Women. Journal of Environmental Psychology, Volume 15, Issue 4 (March 1995), pages 23-38. Elsevier.

Dey, A. (2001). Understanding and Using Context. *Personal and Ubiquitous Computing*, Volume 5, No. 1 (February 2001), pages 4-7. Berlin, Heidelberg: Springer.

Dey, A. and Abowd, G. (2000). CybreMinder: A Context-Aware System for Supporting Reminders. In *Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing* (HUC), pages 172-186. Berlin, Heidelberg: Springer.

Dix, A. (2002). Beyond Intention - Pushing Boundaries With Incidental Interaction. In *Proceedings of Building Bridges: Interdisciplinary Context-Sensitive Computing*, Glasgow University.

Dix, A. et al. (2000). Exploiting Space and Location as a Design Framework for Interactive Mobile Systems. ACM Transactions on Information Systems (TOIS), Volume 7, Issue 3, pages 285-321. New York, NY: ACM.

Dougherty, J. and Keller, C. (1985). Taskonomy: A Practical Approach to Knowledge Structures. In: J. Dougherty (Ed.), *Directions in Cognitive Anthropology*, pages 161-174. Urbana, IL: University of Illinois.

Dürr, F. and Rothermel, K. (2003). On a Location Model for Fine-Grained Geocast. In *Proceedings of the 5th International Conference on Ubiquitous Computing* (UBICOMP 2003), pages 18-35. Berlin, Heidelberg: Springer.

Farrell, M. J., Arnold, P., Pettifer, S., Adams, J., Graham, T. and MacManamon, M. (2003). Transfer of Route Learning from Virtual to Real Environments. *Journal of Experimental Psychology: Applied*, 9 (4), pages 219-227. Washington, DC: American Psychological Association (APA).

Fensel, D, Hendler, J. Lieberman, H. and Wahlster, W. (Eds.) (2003). Spinning the Semantic Web. Bringing the World Wide Web to Its Full Potential. Cambridge, MA: MIT.

Finkenzeller, D. (2008). Modellierung komplexer Gebäudefassaden in der Computergraphik. Dissertation, Fakultät für Informatik, Institut für Betriebs- und Dialogsysteme (IBDS), Universitätsverlag Karlsruhe. http://digbib.ubka.uni-karlsruhe.de/volltexte/1000007847

Floerkemeier, C., Langheinrich, M., Fleisch, E., Mattern, F. and Sarma, S.E. (Eds.) (2008). *The Internet of Things. First International Conference* (IOT 2008, Zurich, Switzerland), March 26-28, LNCS 4952, Berlin, Heidelberg: Springer.

Fox, A., Johanson, B., Hanrahan, P. and Winograd, T. (2000). Integrating Information Appliances into an Interactive Workspace. *IEEE Computer Graphics and Applications*, Volume 20 (3), pages 54-65.

Freeman, E., Hupfer, S. and Arnold, K. (1999). JavaSpaces: Principles, Patterns and Practice. Addison Wesley.

Gaisbauer, C. and Frank, A. (2008). A Wayfinding Model for Pedestrian Navigation. In *Online Proceedings of the 11th AGILE International Conference on Geographic Information Science – Taking Geoinformation Science One Step Further* (AGILE 2008, University of Girona, Catalonia, Spain, May 5-8), LNG&C, pages 1-9. Berlin, Heidelberg: Springer.

Gapp, K.-P. (1997). Objektlokalisation: ein System zur sprachlichen Raumbeschreibung. Wiesbaden, Germany: Deutscher Universitätsverlag.

Gaver, B., Dunne, T. and Pacenti, E. (1999). Design: Cultural probes. *interactions*, Volume 6, No. 1 (January 1999), pages 21-29. New York, NY: ACM.

Gay, G. and Hembrooke, H. (2004). Activity-Centered Design - An Ecological Approach to Designing Smart Tools and Usable Systems. Cambridge, MA: MIT, 2004, ISBN 0-262-07248-3.

Gelernter, D. and Carreiro, N. (1992). Coordination Lang¬uages and their Significance. *Communications of the ACM*, Volume 35, Issue 2 (February 1992), pages 97-107.

Geography Markup Language (GML) 2.0, OpenGIS Implementation Specification, OGC Document Number: 01-029, 20 February 2001. Website: http://www.opengis.net/gml/01-029/GML2.html

Geven, A., Buiza, C., Obrist, M., Reitberger, W. and Tscheligi, M. (Eds.) (2008). *Capturing Ambient Assisted Living Needs*. Workshop at European Conference on Ambient Intelligence (Ami2008, Nürnberg, Germany, November 19), In Online Proceedings: http://workshops.icts.sbg.ac.at/ami2008aal/

Ghazi-Zahedi, K., Laue, T., Röfer, T., Scholl, P., Spiess, K., Twickel, A. and Wischmann, S. (2005). RoSiML - Robot Simulation Markup Language. http://www.informatik.uni-bremen.de/spprobocup/RoSiML.html

Gottfried, B., Guesgen, H. W. and Hübner, S. (2006). Spatiotemporal Reasoning for Smart Homes. In: J.C. Augusto and C. Nugent (Eds.), *Designing Smart Homes – The Role of Artificial Intelligence*. LNCS 4008, pages 16-34. Berlin, Heidelberg: Springer.

Gruber, T. R. (1993). Toward Principles for the Design of Ontologies used for Knowledge Sharing. Presented at the Padua workshop on Formal Ontology, March 1993, later published in International Journal of Human-Computer Studies, Volume 43, Issues 4-5 (November 1995), pages 907-928. Elsevier.

Hao Hu, D., Pan, S. J., Zheng, V. W., Liu, N. N. and Yang, Q. (2008). Real World Activity Recognition With Multiple Goals. In *Proceedings of the 10th international Conference on Ubiquitous Computing* (UbiComp '08, Seoul, Korea), September 21-24), Volume 344, pages 30-39. New York, NY: ACM.

Harrison, S. and Dourish, P. (1996). Re-place-ing space: the roles of place and space in collaborative systems. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work* (CSCW '96, Boston, Massachusetts, United States, November 16-20), pages 67-76. New York, NY: ACM.

Hart, P., Nilsson, N. and Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. In *IEEE Transactions on Systems Science and Cybernetics*, Volume 4, No. 2, pages 100-107, 1968.

Heckmann, D. (2003). Introducing Situational Statements as an Integrating Data Structure for User Modeling, Contextawareness and Resource-Adaptive Computing. In: A. Hoto and G. Stumme (Eds.), *LLWA Lehren - Lernen - Wissen - Adaptivität* (ABIS2003), pages 283-286. Karlsruhe, Germany.

Heckmann, D. (2005). Ubiquitous User Modeling. DISKI 297. Berlin: Akademische Verlagsgesellschaft Aka.

Heckmann, D. and Loskyll, M. (2009). UbisLabel & UnderscoreEncoding: a new Approach for Label-Encoding in the Multilingual World Wide Web. In *Proceedings of the IADIS Interational Conference Internet/WWW 2009*. (in print)

Heckmann, D., Loskyll, M., Math, R., Recktenwald, P. and Stahl, C. (2009). UbisWorld 3.0: a Semantic Tool Set for Ubiquitous User Modeling. Demonstration Description, In *Online Proceedings of First International Conference on User Modeling, Adaptation, and Personalization* (UMAP 2009).

Heider, Th. and Kirste, Th. (2002). Supporting Goal Based Interaction With Dynamic Intelligent Environments, In: van Harmelen, F. (Ed.), *Proceedings of the 15th European Conference on Artificial Intelligence* (ECAI'2002, Lyon, France, July 22-26), pages 596-600. IOS.

Hemmert, F. (2008). Ambient Life: Permanent Tactile Life-like Actuation as a Status Display in Mobile Phones. In *Adjunct Proceedings of the 21st annual ACM symposium on User Interface Software and Technology* (UIST'08, Monterey, California, USA, October 20-22). New York, NY: ACM.

Herfet, T., Kirste, T. and Schnaider, M. (2001). EMBASSI: Multimodal Assistance for Infotainment and Service Infrastructures. *Computers & Graphics*, Volume 25, No. 4, pages 581–592. Elsevier.

Hightower, J. and Borriello, G. (2001). Location systems for Ubiquitous Computing. *IEEE Computer*, Volume 34, Issue 8 (August 2001), pages 57-66. Washington, DC: IEEE Computer Society.

Hoffmann, A. (2008). Realisierung einer Kaufhausnavigation. Bachelor Thesis, Department of Computer Science, Saarland University.

Holland, S., Morse, D. and Gedenryd, H. (2002). AudioGPS: Spatial Audio Navigation with a Minimal Attention Inter¬face. *Personal Ubiquitous Computing*, Volume 6, No. 4 (January 2002), pages 253-259. Berlin, Heidelberg: Springer.

Höllerer, T., Feiner, S., Terauchi, T., Rashid, G. and Hallaway, D. (1999). Exploring MARS: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System. *Computers & Graphics*, Volume 23, No. 6 (December 1999), pages 779-785. Elsevier.

Holman, D. and Vertegaal, R. (2008). Organic User Interfaces: Designing Computers in any Way, Shape, or Form. *Communications of the ACM*, Volume 51, No. 6, pages 48-55. New York, NY: ACM.

Hölscher, C., Meilinger, T., Vrachliotis, G. Brösamle, M. and Knauff, M. (2006). Up the Down Staircase: Wayfinding Strategies and Multi-Level Buildings. *Journal of Environmental Psychology*, Volume 26(4), pages 284-299. Elsevier.

Horvitz, E. (1999). Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: the CHI Is the Limit* (CHI '99, Pittsburgh, Pennsylvania, United States, May 15 - 20), pages 159-166. New York, NY: ACM.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D. and Rommelse, K. (1998). The Lumière project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In: G. F. Cooper and S. Moral (Eds.), *Uncertainty in Artificial Intelligence: Proceedings of the 14th Conference*, pages 256-265. San Francisco, CA: Morgan Kaufmann.

Horvitz, E., Koch, P. and Subramani, M. (2007). Mobile Opportunistic Planning: Methods and Models. In *Proceedings of the 11th International Conference on User Modeling* (UM 2007, Corfu, Greece), June 25-29, pages 228-237. Berlin, Heidelberg: Springer.

Huang, E., Russel, D. and Sue, A. (2004). IM Here: Public Instant Messaging on Large, Shared Displays for Workgroup Interactions. In *Proceedings of CHI 2004*, pages 279-286. New York, NY: ACM. 2004.

Hunt, E. and Walter, D. (1999). Orientation and Wayfinding: A Review, Tech. Rep. Nr. N00014-96-0380, Arlington, VA: Office of Naval Research.

Inkpen, K., Hawkey, K., Kellar, M., Mandryk, R., Parker, K., Reilly, D., Scott, S. and Whalen, T. (2005).

Exploring Display Factors that Influence Co-Located Collaboration: Angle, Size, Number, and User Arrangement. In *Proceedings of the 11th International Conference on Human-Computer Interaction* (HCI International 2005, Las Vegas, USA, July 22-27). Hillsdale, NJ: Lawrence Erlbaum Associates.

Ishii, H. (2008a). Tangible Bits: Beyond Pixels. In *Proceedings of the 2nd International Conference on Tangible and Embedded interaction* (TEI'08, Bonn, Germany, February 18-20), pages xv-xxv. New York, NY: ACM.

Ishii, H. (2008b). The Tangible User Interface and its Evolution. *Communications of the ACM*, Volume 51, No. 6, pages 32-36. New York, NY: ACM.

ISO/IEC 24752 (2006), Information Technology – User Interfaces – Universal Remote Console – 5 parts. International Organization for Standardization (ISO), 2006.

ISTAG 2001: ISTAG - Scenarios for Ambient Intelligence, ftp://ftp.cordis.lu/pub/ist/docs/istagscenarios2010.pdf

ISTAG 2003: ISTAG - Ambient Intelligence: from vision to reality, ftp://ftp.cordis.lu/pub/ist/docs/istag-ist2003_consolidated_report.pdf

Iwabuchi, E., Nakagawa, M. and Siio, I. (2009). Smart Makeup Mirror: Computer-Augmented Mirror to Aid Makeup Application. In *Proceedings of 13th International Conference on HCI* (HCI International 2009, San Diego, CA, USA), July 19-24, LNCS 5613, pages 495-503. Berlin, Heidelberg: Springer.

Jameson, A. (2006). Adaptive Interfaces and Agents. In: J. A. Jacko and A. Sears (Eds.), *Human-Computer Interaction Handbook,* 2nd Edition. Mahwah, NJ: Lawrence Erlbaum Associates.

Jiang, C. and Steenkiste, P. (2002). A Hybrid Location Model with a Computable Location Identifier for Ubiquitous Computing. In *Proceedings of the 4th International Conference on Ubiquitous Computing* (UBICOMP 2002, Göteborg, Sweden), pages 246-263. Berlin, Heidelberg: Springer.

Johanson, B. and Fox, A. (2002). The Event Heap: A Coordination Infrastructure for Interactive Workspaces. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications*, page 83. Washington, DC: IEEE Computer Society.

Kaptelini, V. and Nardi, B. (2006). Acting with Technology – Activity Theory and Interaction Design. Cambridge, MA: MIT.

Katz, S., Ford, A.B., Moskowitz, R.W., Jackson, B.A. and Jaffe, M.W. (1963). Studies of Illness in the Aged: The Index of ADL: A Standardized Measure of Biological and Psychosocial Function. *Journal of the American Medical Association* (JAMA), Volume 185, Issue 12 (September 1963), pages 914-919. Philadelphia, PA: American Medical Publishers Association.

Kindberg, T. Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B., Spasojevic, M. (2002). People, Places, Things: Web Presence for the Real World. *Mobile Networks and Applications*, Volume 7, Issue 5, pages 365-376. Cologne, Germany: Kluwer.

Kirste, H., Herfet, T. and Schnaider, M. (2001). EMBASSI: Multimodal Assistance for Universal Access to Infotainment and Service Infrastructures. In *Proceedings of the 2001 EC/NSF Workshop on Universal Accessibility of Ubiquitous Computing* (WUAUC), pages 356-361. Morristown, NJ: Association for Computational Linguistics.

Klippel, A., Richter, K.-F., Barkowsky, T. and Freksa, C. (2004). The Cognitive Reality of Schematic Maps. In: A. Zipf, T. Reichenbacher, L. Meng (Eds.), *Map-Based Mobile Services - Theories, Methods and Implementations*. Berlin, Heidelberg: Springer.

Koile, K., Tollmar, K., Demirdjian, D., Shrobe, H. and Darrell, T. (2003). Activity Zones for Context-Aware Computing. In *Proceedings of Ubiquitous Computing, 5th International Conference* (Ubicomp'03, Seattle), pages 90-106. Berlin, Heidelberg: Springer.

Kray, C. (2003). Situated Interaction on Spatial Topics. DISKI 274. Berlin: Akademische Verlagsgesellschaft Aka.

Kray, C., Wasinger, R. and Kortuem, G. (2004). Concepts and Issues in Interfaces for Multiple Users and Multiple Devices. In *Proceedings of the Workshop on Multi-User and Ubiquitous User Interfaces* (MU3I), at IUI 2004, Funchal, Madeira, Portugal, SFB 378 Memo Nr. 83, pages 7-11. ISSN 0944-7822.

Kray, C., Cheverst, K., Fitton, D., Sas, C., Patterson, J., Rouncefield, M. and Stahl, C. (2006). Sharing Control of Dispersed Situated Displays Between Nomandic and Residential Users. In *Proceedings of the 8th Conference on Human-Computer interaction with Mobile Devices and Services* (MobileHCI '06, Helsinki, Finland, September 12-15), pages 61-68. New York, NY: ACM.

Kray, C., Cheverst, K., Harrison, M., Hamhoum, F. and Wagner, J. (2008). Towards a Location Model for Indoor Navigation Support Through Public Displays and Mobile Devices. *Workshop on Mobile Interaction with the Real World* (MIRW) at Mobile HCI 2008.

Krieg-Brückner, B., Röfer, T., Carmesin, H.-O. and Müller, R. (1998). A Taxonomy of Spatial Knowledge for Navigation and its Application to the Bremen Autonomous Wheelchair. In: C. Freksa, C. Habel and K.F. Wender (Eds.), *Spatial Cognition. An Interdisciplinary Approach to Representing and Processing Spatial Knowledge*, pages 373-397. Berlin, Heidelberg: Springer.

Kröner, A., Heckmann, D. and Wahlster, W. (2006). SPECTER: Building, Exploiting, and Sharing Augmented Memories. In K. Kogure (Ed.), *Workshop on Knowledge Sharing for Everyday Life 2006* (KSEL06, Kyoto, Japan), pages 9-16. ATR Media Information Science Laboratories. ISBN 4-902401-03-7.

Krüger, A., Butz, A., Müller, C., Stahl, C., Wasinger, R., Steinberg, K.-E. and Dirschl, A. (2004). The Connected User Interface: Realizing a Personal Situated Navigation Service. In *Proceedings of the 9th International Conference on Intelligent User Interfaces* (IUI 2004), pages 161-168. New York, NY: ACM.

Kruppa, M., Krüger, A., Rocchi, R., Stock, O. and Zancanaro, M. (2003). Seamless Personalized TV-Like Presentations on Mobile and Stationary Devices in a Museum. In *Proceedings of the Seventh International Cultural Heritage Informatics Meeting* (ICHIM 2003, Paris, September 8-12), pages 1-19. Toronto, Canada: Archives & Museum Informatics Europe.

Kruppa, M., Spassova, L. and Schmitz, M. (2005). The Virtual Room Inhabitant. In *Proceedings of 2nd Workshop on Multi-User and Ubiquitous User Interfaces* (MU3I), In Conjunction with the International Conference on Intelligent User Interfaces (IUI 2005, San Diego, CA, USA), SFB 378 Memo Nr. 85, ISSN 0944-7822, pages 1-2.

Kruppa, M. (2006). Migrating Characters: Effective User Guidance in Instrumented Environments. DISKI 301. Berlin: Akademische Verlagsgesellschaft Aka.

Lashkari, Y., Metreal, M. and Maes, P. (1994). Collaborative Interface Agents. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 444-449. Cambridge, MA: MIT.

Lassila, O. and Swick, R. R. (1999). Resource Description Framework (RDF) Model and Syntax Specification, W3C. W3C Recommendation.

Leonhardt, U. (1998). Supporting Location-Awareness in Open Distributed Systems. PhD thesis, Imperial College London, Department of Computing.

Leontiev, A. (1978). Activity, Consciousness, and Personality. Englewood Cliffs, N.J.: Prentice-Hall. (Original work published in Russian in 1975).

Leontiev, A. (1979). The Problem of Activity in Psychology. In: J. Wertsch (Ed.), The Concept of Activity in Soviet Pyschology. Armonk, NY: M.E. Sharpe, Inc.

Lieberman, H. and Espinosa, J. (2006). A Goal-Oriented Interface to Consumer Electronics Using Planning and Commonsense Reasoning. In *Proceedings of the 11th International Conference on Intelligent User Interfaces* (IUI 2006, Sydney, Australia), pages 226-233. New York, NY: ACM.

Lifton, J. (2007). Dual Reality: An Emerging Medium. PhD thesis at the School of Architecture and Planning, Massachusetts Institute of Technology (MIT), USA.

Loskyll, M. (2009). Ontology Development in the Upcoming Web 3.0 Era: Design and Implementation of

Visualization, Search and Online Editing Tools. Master Thesis, Dept. of Computer Science, Saarland University.

Loskyll, M., Heckmann, D. and Kobayashi, I. (2009). UbisEditor 3.0: Collaborative Ontology Development on the Web. In *Proceedings of the Workshop on Web 3.0: Merging Semantic Web & Social Web (SW)²*, held in conjunction with the 20th ACM Conference on Hypertext and Hypermedia (Hypertext 2009, Torino, Italy).

Maaß, W. (1993). A Cognitive Model for the Process of Multimodal, Incremental Route Descriptions. In: A. U. Frank and I. Campari (Eds.), Spatial Information Theory: A Theoretical Basis for GIS, LNCS 716, pages 1-13. Berlin, Heidelberg: Springer.

Maaß, W. (1996). Von visuellen Daten zu inkrementellen Wegbeschreibungen in dreidimensionalen Umgebungen: Das Modell eines kognitiven Agenten. Dissertation, Saarland University.

Maes, P. (1994). Agents That Reduce Work and Information Overload. *Communications of the ACM*, Volume 37, Issue 7 (July 1994), pages 30-40. New York, NY: ACM.

Makino, R., Kobayashi, I., Izumi, N. and Hasida, K. (2009). An Ontology Approach to Creating a New Recipe by Reusing Parts of the Existing Recipes. In *Proceedings of IEEE International Conference on Intelligent Computing and Intelligent Systems* (ICIS 2009, Shanghai, China, November 20-22). (to appear)

Manola, F. and Miller, E. (Eds.) (2004). RDF Primer. W3C Recommendation, 10 February 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

Meixner, G., Seissler, M. and Nahler, M. (2009). Udit - A Graphical Editor for Task Models. In *Proceedings of the Workshop on Model Driven Development of Advanced User Interfaces* (MDDAUI '09, Sanibel Island, USA), February 8, CEUR Workshop Proceedings, Volume 439. ISSN 1613-0073. http://CEUR-WS.org/Vol-439/

Montello, D. R. (1993). Scale and Multiple Psychologies of Space. In: A. U. Frank and I. Campari (Eds.), *Spatial Information Theory: A Theoretical Basis for GIS* (COSIT '93), LNCS 716, pages 312-321. Berlin, Heidelberg: Springer.

Müller, C. (2005). Zweistufige Kontextsensitive Sprecherklassifikation am Beispiel von Alter und Geschlecht. Dissertation, Dept. of Computer Science, Saarland University.

Müller, C. (Ed.) (2007). Speaker Classification I. Fundamentals, Features, and Methods. LNCS 4343. Berlin, Heidelberg: Springer.

Münzer, S. and Stahl, C. (2007). Providing Individual Route Instructions for Indoor Wayfinding in Complex, Multi-Level Buildings. In: F. Probst and C. Keßler (Eds.), GI-Days 2007 - Young Researchers Forum (September 10-12, Münster, Germany), pages 241-246. IfGI Prints Nr. 30. Solingen: Verlag Natur & Wissenschaft.

Münzer, S. and Stahl, C. (2008). Learning of Visual Route Instructions for Indoor Wayfinding. In: Hölscher, C. (Ed.) *Spatial Cognition 2008 Poster Proceedings. International Conference on Spatial Cognition* (Sept. 15-19, Freiburg, Germany). SFB/TR8 Report No. 016-08/2008. University of Bremen / University of Freiburg.

Mutafchiev, S. (2008). Hybride Navigationsvisualisierung auf Mobilen Geräten. Diplomarbeit, Dept. of Computer Science, Saarland University.

Nehmer, J., Becker, M., Karshmer, A. and Lamm, R. (2006). Living Assistance Systems – An Ambient Intelligence Approach, In *Proceedings of the 28th International Conference on Software Engineering* (ICSE, Shanghai, China), pages 43-50. New York, NY: ACM.

Newell, A. (1962). Some Problems on the Basic Organization in Problem-Solving Programs. In: Yovits, M.C., Jacobi, G.T. and Goldstein, G.D. (Eds.), In *Proceedings of the 2nd Conference on Self-Organizing Systems*, pages 393-423. Spartan Books.

Norbisrath, U. and Mosler, C. (2006). Functionality Configuration for eHome Systems. In *Proceedings of the 16th International Conference on Computer Science and Software Engineering* (CASCON'06, Toronto, Canada, October 16-19), New York, NY: ACM.

Norman, D. A. (1983). Some Observations on Mental Models. In: Gentner, D. and Stevens, A. L. (Eds.), *Mental Models*, pages 7-14. Hillsdale, NJ: Lawrence Erlbaum Associates.

Norman, D. A. (1986). Cognitive Engineering. In: Norman, D. and Draper, S (Eds.), *User Centered System Design*, pages 31-61. Hillsdale, NJ: Lawrence Erlbaum Associates.

Norman, D. A. (1988). The Psychology of Everyday Things. New York: Basic Books.

Norman, D. A. (1991). Cognitive Artifacts. In J. M. Carroll (Ed.), *Designing Interaction: Psychology at the Human-Computer Interface*, pages 17-38. New York, NY: Cambridge University Press.

Norman, D. A. (2005). Human Centered Design Considered Harmful. *interactions*, Volume 12, Issue 4 (July and August, 2005), pages 14-19. New York, NY: ACM.

Norman, D. A. (2006). Logic Versus Usage: the Case for Activity-Centered Design. *interactions*, Volume 13, Issue 6 (November 2006), pages 45-46. New York, NY: ACM.

Norman, D. A. (2007). Three Challenges for Design. *interactions*, Volume 14, Issue 1 (January and February 2007), pages 45-46. New York, NY: ACM.

O'Hara, K., Perry, M., Churchill, E. and Russel, D. (2003). Public and Situated Displays: Social and Interactional Aspects of Situated Display Technologies, The Netherlands: Kluwer. ISBN 1-4020-1677-8.

O'Reilly, T. (2005). What Is Web 2.0 – Design Patterns and Business Models for the Next Generation of Software. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1

Oviatt, S. L. (1996). Multimodal interfaces for dynamic interactive maps. In *Proceedings of Conference on Human Factors in Computing Systems* (CHI '96, Vancouver, Canada, April 13-18), pages 95-102. New York, NY: ACM.

Oviatt, S. L. (1997). Multimodal Interactive Maps: Designing for Human Performance, *Human-Computer Interaction: Special Issue on Multimodal interfaces*, Volume 12, No. 1, pages 93-129. Hillsdale, NJ: Lawrence Erlbaum Associates.

Paternò, F. (1999). Model-Based Design and Evaluation of Interactive Applications. Berlin, Heidelberg: Springer.

Paternò, F. (2000). Model-based Design of Interactive Applications. *Intelligence*, Volume 11, Issue 4 (December 2000), pages 26-38. New York, NY: ACM.

Pederson, T. (2003). From Conceptual Links to Causal Relations — Physical-Virtual Artifacts in Mixed-Reality Space. PhD thesis, Dept. of Computing Science, Umeå University, report UMINF-03.14, ISSN 0348-0542, ISBN 91-7305-556-5. Permanent URL: http://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-137

Pierson, J., Jacobs, A., Dreessen, K., Van den Broeck, I., Lievens, B. and Van den Broeck, W. (2006). Walking the Interface: Uncovering Practices Through 'Proxy Technology Assessment', In *Proceedings of Ethnographic Praxis in Industry Conference* (EPIC 2006, organised by Microsoft and Intel, USA), pages 29-40. American Anthropological Association.

Raubal, M. (2002). Wayfinding in Built Environments - The Case of Airports. In: W. Kuhn and U. Streit (Eds.), IfGI Prints Band 14. Solingen: Verlag Natur & Wissenschaft. ISBN 3-927889-01-6.

Rekimoto, J. (2008). Organic Interaction Technologies: From Stone to Skin. *Communications of the ACM*, Volume 51, No. 6, pages 38-44. New York, NY: ACM.

Reuther, A. (2003). useML – Systematische Entwicklung von Maschienenbediensystemen mit XML, Dissertation, Fortschritts-Berichte pak Band 8, Kaiserslautern.

RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee, R. Fielding and L. Masinter, IETF, August 1998. Document URL: http://www.isi.edu/in-notes/rfc2396.txt.

Richardson, T. (2005). The RFB Protocol, Version 3.8, RealVNC Ltd., Website:

http://www.realvnc.com/docs/rfbproto.pdf

Richter, K.-F. and Klippel, A. (2005). A Model for Context-Specific Route Directions. In: C. Freksa, M. Knauff, B. Krieg-Brückner, B. Nebel, and Barkowsky, T. (Eds.), *Spatial Cognition IV. Reasoning, Action, Interaction: International Conference Spatial Cognition 2004,* pages 58-78. Berlin, Heidelberg: Springer.

Riss, U., Rickayzen, A., Maus, H. and van der Aalst, W. (2005). Challenges for Business Process and Task Management. *Journal of Universal Knowledge Management. Special Issue on Knowledge Infrastructures for the Support of Knowledge Intensive Business Processes*, No. 2, pages 77-100.

Royce, W. W. (1987). Managing the Development of Large Software Systems: Concepts and Techniques. In *Proceedings of the 9th International Conference on Software Engineering* (Monterey, California, United States), pages 328-338. Washington, DC: IEEE Computer Society.

Salber, D., Dey, A. K. and Abowd, G. D. (1999). The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Proceedings of CHI 1999,* pages 434-441. New York, NY: ACM.

Schilit, B. and Theimer, M. (1994). Disseminating Active Map Information to Mobile Hosts. IEEE Network, Volume 8(5), pages 22-32.

Schmidt, A. (2000). Implicit Human-Computer Interaction through Context. *Personal Technologies*, Volume 4(2&3), pages 191-199. Berlin, Heidelberg: Springer.

Schmitz, M., Baus, J. and Dörr, R. (2008). The Digital Sommelier: Interacting with Intelligent Products. In *Proceedings of Internet of Things 2008* (IOT 2008, March 26-28, 2008, Zurich, Switzerland), pages 247-262. LNCS 4952. Berlin, Heidelberg: Springer.

Schneider, M. (2003). A Smart Shopping Assistant Utilizing Adaptive Plan Recognition. In: A. Hoto and G. Stumme (Eds.), *Lehren - Lernen - Wissen - Adaptivität* (LLWA04), pages 331-334.

Schneider, M. (2007). The Semantic Cookbook: Sharing Cooking Experiences in the Smart Kitchen. In *Proceedings of the 3rd International Conference on Intelligent Environments* (IE-2007, September 24-25, Ulm, Germany), pages 416-423. Herts, UK: The Institution of Engineering and Technology (IET).

Schneider, M. (2009). Plan Recognition in Instrumented Environments. In *Proceedings of the 5th International Conference on Intelligent Environments* (IE09, Barcelona, Spain), pages 295-302. IOS.

Schneider, M. and Kröner, A. (2008). The Smart Pizza Packing: An Application of Object Memories. In *Proceedings of the 4th International Conference on Intelligent Environments* (IE08, Seattle, WA, USA), July 21-22, pages 1-8. Washington, DC: IEEE Computer Society.

Schwartz, T., Brandherm, B. and Heckmann, D. (2005). Calculation of the User-Direction in an Always Best Positioned Mobile Localization System. In *Proceedings of the International Workshop on Artificial Intelligence in Mobile Systems* (AIMS 2005) in Conjunction with MobileHCI 2005, Salzburg, Austria, 2005.

Shelton, A. L. and McNamara, T. P. (2004). Orientation and perspective dependence in route and survey learning. *Journal of Experimental Psychology: Learning, Memory and Cognition*, Volume 30 (1), pages 158-170. Washington, USA: American Psychological Association (APA).

Siegel, A. W. and White, S. H. (1975). The development of spatial representations of large-scale environments. In: H.W. Reese (Ed.), *Advances in Child Development and Behavior*, Volume 10, pages 9-55. New York, NY: Academic Press.

Siems, U., Herwig, C. and Röfer, T. (1994). SimRobot, ein System zur Simulation Sensorbestückter Agenten in Einer Dreidimensionalen Umwelt. In: B. Krieg-Brückner, G. Roth and H. Schwegler (Eds.), ZKW Bericht, No. 1/94. Zentrum für Kognitionswissenschaften. Universität Bremen.

Smith, D., Irby, C., Kimball, R. Verplank, B. and Harslem, E. (1982). Designing the Star User Interface. *BYTE: The Small Systems Journal*. Volume 7(4), pages 242-82. Peterborough, NH: BYTE Publishing Inc.

Sorrows, M. and Hirtle, S. (1999). The Nature of Landmarks for Real and Electronic Spaces. In: C. Freksa and D. M. Mark (Eds.), *Spatial Information Theory. Cognitive and Computational Foundations of Geographic*

Information Science (COSIT'99), LNCS 1661, pages 37-50. Berlin, Heidelberg: Springer.

Sponselee, A., Schouten, B.A.M. and Bouwhuis, D.G. (2008). Analysing Users' Needs for Telecare: Several Case Studies. Position paper, presented at *Capturing Ambient Assisted Living Needs Workshop* at Ami2008 Conference on Ambient Intelligence, Nürnberg, Germany.

Stahl, C., Baus, J., Krüger, A., Heckmann, D., Wasinger, R. and Schneider, M. (2004). REAL: Situated Dialogues in Instrumented Environments. In: S. Tanimoto, A. Quigley (Eds.), *Workshop on Invisible & Transparent Interfaces*, held at International Conference on Advanced Visual Interfaces (AVI 2004, Gallipoli, Italy), pages 10-15.

Stahl, C. and Heckmann, D. (2004a). Using Semantic Web Technology for Ubiquitous Hybrid Location Modeling. In *1st Workshop on Ubiquitous GIS*, in conjunction with 12th International Conference on Geoinformatics. Gävle, Sweden, 2004.

Stahl, C. and Heckmann, D. (2004b). Using Semantic Web Technology for Ubiquitous Location and Situation Modeling. *The Journal of Geographic Information Sciences*, Volume 10, No. 2 (December 2004), pages 157-165. Berkeley, CA: CPGIS.

Stahl, C., Baus, J., Brandherm, B., Schmitz, M. and Schwartz, T. (2005). Navigational- and Shopping Assistance on the Basis of User Interactions in Intelligent Environments. In *Proceedings of the IEE International Workshop on Intelligent Environments* (IE 2005, June 29, 2005, University of Essex, Colchester, UK), pages 182-191. London, UK: IEE.

Stahl, C., Schmitz, M., Krüger, A. and Baus, J. (2005). Managing Presentations in an Intelligent Environment. In *Proceedings of 2nd Workshop on Multi-User and Ubiquitous User Interfaces 2005* (MU3I 2005), In conjunction with the International Conference on Intelligent User Interfaces (IUI 2005), pages 3-4, SFB 378 Memo Nr. 85. ISSN 0944-7822.

Stahl, C. and Haupert, J. (2006). Taking Location Modelling to new Levels: A Map Modelling Toolkit for Intelligent Environments. *2nd International Workshop on Location- and Context-Awareness*. In: M. Hazas, J. Krumm and T. Strang (Eds.): *LoCA 2006*, LNCS 3987, pages 74-85, 2006. Berlin, Heidelberg: Springer.

Stahl, C. (2006a). Towards a Notation for the Modeling of User Activities and Interactions Within Intelligent Environments. *3rd International Workshop on the Tangible Space Initiative* (TSI 2006). In: T. Strang, V. Cahill, A. Quigley (Eds.), *Pervasive 2006 Workshop Proceedings*, pages 441-452. Deutsches Zentrum für Luft- und Raumfahrt. ISBN 978-3-00-018411-6.

Stahl, C. and Haupert, J. (2006b). Simulating and Evaluating Public Situated Displays in Virtual Environment Models. In: SFB 378 - Resource-Adaptive Cognitive Processes, Memo Nr. 86: T. Pederson, H. Pinto, M. Schmitz, C. Stahl, and L. Terrenghi (Eds.): *International Workshop on Modelling and Designing User Assistance in Intelligent Environments* (MODIE 2006), pages 32-35, 2006. Saarland University. ISSN 0944-7822.

Stahl, C., Pinto, H., Pederson, T., Schmitz, M. and Terrenghi, L. (2006c). MODIE 2006: Modeling and Designing User Assistance in Intelligent Environments. In *Proceedings of the 8th Conference on Human-Computer interaction with Mobile Devices and Services* (MobileHCI '06, Helsinki, Finland, September 12 - 15, 2006), Volume 159, pages 297-298. New York, NY: ACM.

Stahl, C., Heckmann, D., Schwartz, T. and Fickert, O. (2007). Here and Now: A User-Adaptive and Location-Aware Task Planner. In *Proceedings of the International Workshop on Ubiquitous and Decentralized User Modeling* (UbiDeUM 2007) in conjunction with 11th International Conference on User Modeling (UM 2007, Corfu, Greece), pages 52-63.

Stahl, C. (2007). The Roaring Navigator: A Group Guide for the Zoo with Shared Auditory Landmark Display. In *Proceedings of the 9th International Conference on Human Computer interaction with Mobile Devices and Services* (MobileHCI '07, Singapore, September 09-12), pages 383-386, New York, NY: ACM.

Stahl, C. (2008). New Perspectives on Built Environment Models for Pedestrian Navigation. In: C., Hölscher, (Ed.) *Spatial Cognition 2008 Poster Proceedings*. International Conference on Spatial Cognition (September 15-19, Freiburg, Germany). SFB/TR8 Report No. 016-08/2008. Universität Bremen / Universität Freiburg.

Stahl, C., Heckmann, D., Schneider, M. and Kröner, A. (2008). An Activity-Based Approach to the Design of User Assistance in Intelligent Environments. Position paper, presented at *Capturing Ambient Assisted Living Needs Workshop* at Ami2008 Conference on Ambient Intelligence, Nürnberg, Germany.

Stahl, C., Schwartz, T., Baus, J. and Wahlster, W. (2009). Seamless Resource Adaptive Navigation. In: Crocker, M. and Siekmann, J. (Eds). *Resource-Adaptive Cognitive Processes*. Cognitive Technologies Series. Berlin, Heidelberg: Springer. (in print)

Stahl, C. (2009). Modeling and Designing User Assistance in Intelligent Environments with the GOAL Method. In *Workshop on Interaction Techniques and Metaphors in Assistive Smart Environments*, in conjunction with 3rd European Conference on Ambient Intelligence (AmI 2009, Nov. 18, Salzburg, Austria). Berlin, Heidelberg: Springer. (in print)

Starner, T., Mann, S., Rhodes, B., Levine, J., Healey, J., Kirsch, D., Picard, R. and Pentland, A. (1997). Augmented Reality Through Wearable Computing. *Presence: Teleoperators and Virtual Environments*, Volume 6, Issue 4 (December 1997), pages 386-398. Cambridge, MA: MIT.

Stephan, P., Heck, I., Kraus, P. and Frey, G. (2009). Evaluation of Indoor Positioning Technologies Under Industrial Application Conditions in the SmartFactoryKL Based on EN ISO 9283. In *Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing* (INCOM 2009, Moscow, Russian Federation), pages 874-879. ICS / RAS.

Stiehl, W. D., Lee, J. K., Breazeal, C., Nalin, M., Morandi, A. and Sanna, A. (2009). The Huggable: a Platform for Research in Robotic Companions for Pediatric Care. In *Proceedings of the 8th International Conference on interaction Design and Children* (IDC'09, Como, Italy, June 3-5), pages 317-320. New York, NY: ACM.

Stock, O. and Zancanaro, M. (2007). PEACH – Intelligent Interfaces for Museum Visits. Berlin, Heidelberg: Springer.

Storf, H., Becker, M. and Riedl, M. (2009). Rule-based Activity Recognition Framework: Challenges, Technique and Learning. In *ICST Conference PervaSense09: Situation Recognition and Medical Data Analysis*, in conjunction with 3rd International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth 2009, London, UK, March 31), pages 1-7. IEEE Xplore Digital Library.

Streitz, N., Tandler, P., Müller-Tomfelde, C. and Konomi, S. (2001). Roomware: Towards the Next Generation of Human-Computer Interaction Based on an Integrated Design of Real and Virtual Worlds. In: Carroll, J. (Ed.) *Human-Computer Interaction in the New Millenium*, pages 553-578. Reading, MA: Addison-Wesley.

Streitz, N., Prante, T., Röcker, C., van Alphen, D., Stenzel, R., Magerkurth, C., Lahlou, S., Nosulenko, V., Jegou, F., Sonder, F. and Plewe, D. (2007). Smart Artifacts as Affordances for Awareness in Distributed Teams. In: N. Streitz, A. Kameas, I. Mavrommati (Eds.), *The Disappearing Computer*, pages 3-29. LNCS 4500. Berlin, Heidelberg: Springer.

Subasi, Ö. (2008). Optimizing Qualitative User Needs Assessment through Affective Analysis. Position paper, presented at *Capturing Ambient Assisted Living Needs Workshop* at Ami2008 Conference on Ambient Intelligence, Nürnberg, Germany.

Thorndyke, P.W. and Hayes-Roth, B. (1982). Differences in Spatial Knowledge Acquired From Maps and Navigation. *Cognitive Psychology*, Volume 14, Issue 4, pages 560-589. Elsevier.

Tsujita, H., Tsukada, K. and Siio, I. (2008). SyncDecor: Communication Appliances for Virtual Cohabitation. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (AVI 08, Napoli, Italy), May 28-30, pages 449-453. New York, NY: ACM.

Tsukada, K., Tsujita, H. and Siio, I. (2008). TagTansu: A Wardrobe to Support Creating a Picture Database of Clothes. In *Adjunct Proceedings of Pervasive2008* (Sydney, Australia), May 19-22, pages 49-52. Berlin, Heidelberg: Springer.

Vadas, K., Patel, N., Lyons, K., Starner, T. and Jacko, J. (2006). Reading on-the-go: A Comparison of Audio and Hand-Held Displays. In *Proceedings of the 8th Conference on Human-Computer interaction with Mobile Devices and Services* (MobileHCI '06, Helsinki, Finland, September 12-15), Volume 159, pages 219-226. New York, NY: ACM.

Velthuijsen, H. (1992). The Nature and Applicability of the Blackboard Architecture. Proefschrift Maastricht, PTT Research.

Veys, A. (2008). Assessing the Potential of Video Surveillance in Home Care. Position paper, presented at *Capturing Ambient Assisted Living Needs Workshop* at AmI2008 Conference on Ambient Intelligence, Nürnberg, Germany.

Wahlster, W., Hecking, M. and Kemke, Ch. (1988). SC: Ein intelligentes Hilfesystem für SINIX. In: Gollan, B., Paul, W.J., Schmitt, A. (Eds.) *Innovative Informationsinfrastrukturen*. Informatik-Fachberichte 184, pages 81-100. Berlin, Heidelberg: Springer.

Wahlster, W. and Kobsa, A. (1989). User Models in Dialog Systems. In: A. Kobsa and W. Wahlster (Eds.), *User Models in Dialog Systems*, pages 4-34. Berlin, Heidelberg: Springer.

Wahlster, W. (1999). Agent-based Multimedia Interaction for Virtual Web Pages. In *Proceedings of the 4th International Conference on intelligent User interfaces* (IUI '99, Los Angeles, California, USA, January 05 - 08, 1999), page 11. New York, NY: ACM.

Wahlster, W., Reithinger N. and Blocher, A. (2001). SmartKom: Multimodal Communication with a Life-Like Character. In *Proceedings of 7th European Conference on Speech Communication and Technology* (Eurospeech 2001, Aalborg, Denmark, September 2001), Volume 3, pages 1547 - 1550. ICSA Archive.

Wahlster, W. (2003). Towards Symmetric Multimodality: Fusion and Fission of Speech, Gesture, and Facial Expression. In: A. Günter, R. Kruse and B. Neumann (Eds.), *Proceedings of the 26th German Conference on Artificial Intelligence* (KI 2003, Hamburg, Germany, Sept. 2003), LNCS 2821, pages 1-18. Berlin, Heidelberg: Springer.

Wahlster, W. (2007). Digital Product Memory: Embedded Systems Keep a Diary, *Harting's Technology Newsletter tec.News*, Issue 15 (2007), pages 7-9. Harting Technology Group.

Wahlster, W., Kröner, A. and Heckmann, D. (2006). SharedLife: Towards Selective Sharing of Augmented Personal Memories. In: O. Stock and M. Schaerf (Eds.) *Reasoning, Action and Interaction in AI Theories and Systems. Essays Dedicated to Luigia Carlucci Aiello*, LNAI 4155, pages 327-342. Berlin, Heidelberg: Springer.

Want, R., Hopper, A., Falcão, V. and Gibbons, J. (1992). The Active Badge Location System. ACM *Transactions on Information Systems* (TOIS), Volume 10, Issue 1, 1992, pages 91-102. New York, NY: ACM.

Wasinger, R., Oliver, D., Heckmann, D., Braun, B., Brandherm, B. and Stahl, C. (2003). Adapting Spoken and Visual Output for a Pedestrian Navigation System, based on given Situational Statements. In: A. Hoto and G. Stumme (Eds.), *LLWA Lehren – Lernen – Wissen – Adaptivität* (ABIS Karlsruhe, Germany 2003), pages 343-346.

Wasinger, R., Stahl, C. and Krüger, A. (2003b). Robust Speech Interaction in a Mobile Environment Through the use of Multiple and Different Media Types. In *Proceedings of the 8th European Conference on Speech, Communication and Technology* (EUROSPEECH'03, Geneva, Switzerland, September 1-4), pages 1049-1052. ICSA Archive: http://www.isca-speech.org/archive/archive_papers/eurospeech_2003/e03_1049.pdf

Wasinger, R., Stahl, C. and Krüger, A. (2003c). M3I in a Pedestrian Navigation & Exploration System. In *Proceedings of the 5th International Symposium on Human Computer Interaction with Mobile Devices* (Mobile HCI 2003, Udine, Italy, September 8-11), LNCS 2795, pages 481-485. Berlin, Heidelberg: Springer.

Wasinger, R., Schneider, M., Baus, J. and Krüger, A. (2004). Multimodal Interaction with an Instrumented Shelf. In *Workshop on Artificial Intelligence in Mobile Systems 2004* (AIMS 2004, Nottingham, UK), pages 36-43. ISSN 0944-7822.

Weiser, M. (1991). The Computer for the Twenty-First Century. *Scientific American*, Volume 265, Issue 3 (September 1991), pages 94-104. Scientific American, Inc.

Wilensky, R., Arens, Y. and Chin, D. (1984). Talking to UNIX in English: an overview of UC. *Communications* *of the ACM*, Volume 27, Issue 6 (June 1984), pages 574-593.

Winograd, T. (1997). From Computing Machinery to Interaction Design. In: P. Denning and R. Metcalfe (Eds.) *Beyond Calculation: the Next Fifty Years of Computing*, pages 149-162. Berlin, Heidelberg: Springer.

Wooldridge, J. (2000). Reasoning About Reactional Agents. Cambridge, MA: MIT.

Wyckoff, P., McLaughry, S. W., Lehman, T. and Ford, D. A. (1998). T Spaces. *IBM Systems Journal*, Volume 37, No. 3, pages 454-474. IBM Corp.

Zühlke, D. and Thiels, N. (2008). Useware Engineering: A Methodology for the Development of User-friendly Interfaces. *Library Hi Tech*, Volume 26, No. 1 (2008), pages 126-140. Bingley, UK: Emerald Group Publishing Ltd.

Zukerman, I., Albrecht, D. and Nicholson, A. (1999). Predicting Users' Requests on the WWW. In *Proceedings of the 7th International Conference on User Modeling* (UM99, Banff, Canada), pages 275-284. Berlin, Heidelberg: Springer.

A List of Yamamoto Object Properties

Table 18: Polygon properties.

Property	Type	Description
Accessibility	Enumeration	This property defines whether the surface can be accessed by pedestrians, cars, or others. Combinations such as “car, pedestrian” can be entered manually. Polygons which are “not passable” are not used for the route finding algorithm. Use this setting for very small polygons which might cause problems for route finding.
Landmark	Integer	The visual salience of a building or room can be expressed as percentage (0 - 100). This attribute will be used in the future by advanced route finding algorithms in order to create route descriptions that are easier to understand through references to landmark objects. The landmark property must be enabled or disabled for each edge separately (Boolean attribute).
Name	String or UbisName	Any string is allowed to name a polygon, and the special UbisName syntax can be used to connect this model to the symbolic location model of UbisWorld. UbisNames have the schema “namespace..id.label”, e.g. “N..00400347.Kitchen”. Once a valid UbisPointer has been given, the Name attribute will be locked to prevent it from renaming. To unlock the name, remove the UbisName by accessing the polygon’s context menu via the right mouse button.
PolygonType	Enumeration	At the moment, only the types “Elevator” and “Stairs” have a special meaning for route finding. Elevators connect to all polygons with the same “Name” attribute for route planning. Polygons of type “Stairs” are displayed as a sequence of n steps, where n can be set using the “number of steps” attribute. This property will be explained in more detail in the section on 3-D visualization.
Type	Enumeration	The type distinguishes between indoor and outdoor objects. This attribute is not used for route finding at the moment. If “indoor” is selected, Yamamoto automatically sets the 3-D rendering attributes of all edges according to their passableness. Passable edges become doors, all others walls.

Table 19: Point properties.

Property	Type	Description
Point-ID	Integer (read-only)	Each vertex has an unique internal identification number. These successive ids are given to points in the order of creation, beginning from zero, and can’t be changed by the user. In the YML file, they appear as “ID0”, “ID1” etc.
X	Float	Width coordinate in model units. The model units refer to the pixel coordinate space of the backdrop image texture.
Y	Float	Depth coordinate in model units.
Z	Float	Height coordinate, also in model units.
Longitude	Float (read-only)	If the current model is geo-referenced, the model units will be also displayed in geographical coordinates.
Latitude	Float (read-only)	

Table 20: Edge properties.

Property	**Type**	**Description**
Passableness	Enumeration	The passableness of an edge represents the connectivity between adjacent polygons regarding the route finding process. The passableness can be set to “none”, which means that it is impossible to cross this edge in order to move from one polygon to another, or respectively the rooms that are represented by the polygons. Typically, all walls or murals are not passable, so the passableness state of the according edge should be set to “none”. Vice versa, all edges that do not represent obstacles should be set to “both”. Doors for example allow the user to move through them in both directions. In contrary, barriers such as turnstiles only afford pedestrians to cross them in one direction. Therefore the settings “left” and “right” indicate a direction which is indicated by a small green triangle. Please note that internally all edges are directed (from point *A* to point *B*) even though in geometry the edges of a polygon are undirected lines. We purposefully use this feature of our implementation to save an additional point or vector in our data structure.
Label	String	Labels can be used to establish connections between models that are represented in separate *.yml* files. If to edges have the same label, the route finding algorithm assumes that they are two representations of the same feature, such as the entrance of a building. For example, the entrances of a building might be represented in an indoor model, but also in an outdoor model of the campus.
Landmark	Boolean	This boolean flag indicates whether the landmark property of the polygon refers to this edge. For example, only the front of a building might be suitable as a landmark object, but not the back side. In such a case, the polygon object’s Landmark-property would be set to 75%, and only the edge which represents the front side of the building would be set “true”. All other edges would be set to “false”.
Type	Enumeration	Defines the 3-D geometry type based on the following elements: “Open”, “Wall”, “Door”, ”Window”, Mural”, “Handrail”, “WindowWithSkylight”, “DoorWithSkylight”, and “PaintedWall”.
Bottom Edge Height	Float	This value affects the geometry of the “Mural”, “Handrail”, “Window”, and “WindowWithSkylight” elements. The height is interpreted in centimeters. In the XML representation of the model, this value is encoded in the attributes property by using the “i=<*n*>” parameter.
Ceiling Height	Float	This value affects the geometry of the “Wall”, “Door”, and “Window” elements. The height is interpreted in centimeters. In the XML representation of the model, this value is encoded in the attributes property by using the “k=<*n*>” parameter.
Top Edge Height	Float	This value affects the geometry of the “Door” and “Window” elements. The height is interpreted in centimeters. In the XML representation of the model, this value is encoded in the attributes property by using the “j=<*n*>” parameter.
Frame Size	Float	This value affects the width/height of the frame geometry of the “Door” and “Window” elements. In the XML representation of the model, this value is encoded in the attributes property by using the “l=<*n*>” parameter.
Frame Color	String	This value affects the color of the frame geometry of the “Door” and “Window” elements. It is internally represented as RGB (red/green/blue) values between 0 and 255. In the XML representation of the model, this value is encoded in the attributes property by using the “frameColor=<*#RRGGBB>”* parameter.

Segmentation	Integer	This value affects "Door" and "Window" elements. They are rendered with n vertical separator lines so that they appear as a group of 2 or more doors or panes. In the XML representation of the model, this value is appended to the element type, e.g. "Door2" or "DoorWithSkylight2".
Edge Length	Float (read-only)	This value shows the actual length of the edge in cm and model units.
Other Attributes	String	This field shows all unknown parameters of the XML attributes property that could not be interpreted as one of the above.

Table 21: Layer properties.

Property	**Type**	**Description**
Name	String	Layers can be given human-readable names like "Groundfloor" or "Basement" that will be displayed in the layer manager. Internally, Yamamoto uses integer-IDs to identify layers.
Origin X, Y	Integer	This is the origin of the backdrop-image texture within the model coordinate system and should be always set to (0, 0).
Z-Height	Integer	This is the vertical position of the layer. It can be specified either in model units or centimeter. The first floor usually is at height 0 and the height of the second floor layer should equal the wallheight attribute of the first floor layer. Each successive layer should be located at a multiple of the wallheight (E.g. 1st floor at 0 cm, 2nd floor at 250 cm, 3rd floor at 500 cm, etc.)
Length,Width	Integer	The length and width attributes indicate the size of the backdrop image texture in model units. They should equal the image resolution so that one model unit corresponds to one pixel in the image. Sometimes they have to be set manually. If they are set to zero, no backdrop image will be shown.
Background Image	Filename	Chooses the backdrop image texture for this layer and shows a preview. If the image URL is changed, the new texture may not be displayed. Please save and load the model again to see it.
Wall Height	Float	Default height of the walls in this layer. This value can be overridden for some walls in the edge properties.
Door Height	Float	Default height of the door frame in this layer. This value can be overridden for some walls in the edge properties.
Mural Height	Float	Default height of the murals and handrails in this layer. This value can be overridden for some instances in the edge properties.

Table 22: Box properties.

Property	**Type**	**Description**
Color	String	This value defines the color of the box. It is internally represented as RGB (red/green/blue) values. In the XML representation of the model, this value is encoded by using the "color=<*#RRGGBB*>" attribute.
Corner Length	Length	If a corner length greater than zero is specified, the "Corner" shape will have the front right corner diagonally cut off.
Depth	Length	The depth of the box. If the depth is set to zero, only the front image will be shown (and without a frame).
Height	Length	The height of the box. If the height is set to zero, only the top image will be shown (and without a frame).
Image Front	Filename	The image texture that is mapped to the front of the box. In order to be *OpenGL* compliant, the resolution should be square and a power of two. *YAMAMOTO* supports all common file formats, such as *BMP*, *PNG*, *GIF*, and *JPEG*. PNGs also support transparency.

Image Top	Filename	The image texture that is mapped to the top of the box.
Name	String or UbisName	The symbolic name of the box. Any string is valid name, but it is recommended to use an UbisName as reference to physical elements of the UbisWorld ontology.
Orientation	Float	The orientation of the box in degrees between 0 and 360; A value of 0 aligns the box to the Y axis, a value of 90 aligns the box with the X axis. The orientation of a nested box element is relative to its parent.
Thickness	Length	Specifies the thickness of a table's legs and a shelf's corpus.
Type	Enumeration	The type defines the shape of the object. Valid values currently are "Group", "Box", "Shelf", "Table", "Corner", "Storage", and "SlidingDoor". The "Group" shape has no geometry associated with it, it is only used to group other objects.
Width	Length	Specifies the width of the box.
X, Y, Z	Length	Specifies the bottom-center coordinate of the box. The z coordinate refers to the bottom of the box (relative to the layer's z-value) and is zero for objects standing on the ground. The position of a nested box element is relative to its parent.

Table 23: Point, Circle, and Sector item properties.

Property	**Type**	**Description**
Name	String	The name of the item
Info	String	Any additional information, for example the ID of an RFID tag.
Type	Enumeration	Further information about the device that is modeled.
X, Y, Z	Float	Specifies the center of the item in model coordinates.
Radius	Float	Specifies the radius of Circle items.
VX, VY, VZ	Float	Specifies the direction and radius of Sector items.
zAngle	Float	Specifies the opening angle of Sector items in degrees.

Table 24: Display properties.

Property	**Type**	**Description**
Name	String	The name of the item, UbisNames are supported.
Aspect Ratio	Enumeration	4:3 or 16:9
Orientation	Enumeration	Portrait or Landscape mode
X, Y, Z	Float	Specifies the center of the display model coordinates.
Pitch	Float	Vertical rotation of the display in degrees [-90, +90]
Z-Axis Angle	Float	Horizontal rotation of the display in degrees
VNC URL	String	VNC Server IP-address or hostname
VNC Port	String	VNC Server port (default: 5900)
Password	String	VNC Server password (can be provided only here, not manually)

B Yamamoto Document Type Definition

Table 25: Document Type Definition for the Yamamoto YML file format.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Yamamoto Modelling Languange (YML) Doctype Rev 11 edited by Stahl 27/03/2008 -->
<!-- the Model element contains everything -->
<!ELEMENT Model ( ViewPoint*, Scale?, WindRose?, GeoReference?, Avatar?, ObjectTypeDef*,
Coord*, Connection*, Layer+ )>
  <!ATTLIST Model
    name CDATA #REQUIRED
    rootURL CDATA #IMPLIED
    rootObject CDATA #REQUIRED
    xShiftAsSubModel CDATA #REQUIRED
    yShiftAsSubModel CDATA #REQUIRED
    scaleAsSubModel CDATA #REQUIRED
    rotationAsSubModel CDATA #REQUIRED>

<!-- a viewPoint defines a camera perspective in yamamoto -->
<!ELEMENT ViewPoint EMPTY>
  <!ATTLIST ViewPoint
    viewOrthogonal CDATA #REQUIRED
    cameraX CDATA #REQUIRED
    cameraY CDATA #REQUIRED
    cameraAngle CDATA #REQUIRED
    rotation CDATA #REQUIRED
    zoomLevel CDATA #REQUIRED
    activeLayer CDATA #REQUIRED
    name CDATA #REQUIRED>

<!ELEMENT ObjectTypeDef EMPTY>
  <!ATTLIST ObjectTypeDef
    type CDATA #REQUIRED
    color CDATA #REQUIRED>

<!-- scale to compute distances between model-units (pixels) and real length in meter -->
<!ELEMENT Scale EMPTY>
  <!ATTLIST Scale
    x1 CDATA #REQUIRED
    y1 CDATA #REQUIRED
    x2 CDATA #REQUIRED
    y2 CDATA #REQUIRED
    realLength CDATA #REQUIRED>

<!--  geographic  reference  to  convert  model  coordinates  in  geographic  coordinates  of
longitude, latitude and meters above sea level -->
<!ELEMENT GeoReference ( ModelToGeo?, GeoToModel? )>
  <!ATTLIST GeoReference
    axGeo CDATA #REQUIRED
    ayGeo CDATA #REQUIRED
    bxGeo CDATA #REQUIRED
    byGeo CDATA #REQUIRED
    cxGeo CDATA #REQUIRED
    cyGeo CDATA #REQUIRED
    axModel CDATA #REQUIRED
    ayModel CDATA #REQUIRED
    bxModel CDATA #REQUIRED
    byModel CDATA #REQUIRED
    cxModel CDATA #REQUIRED
    cyModel CDATA #REQUIRED
    metersAboveSeaLevel CDATA #REQUIRED>

<!-- ModelToGeo represents a 4x4 matrix M to convert a model coordinate vector into a geo
coordinate vector my multiplication: geo = M * model -->
<!ELEMENT ModelToGeo EMPTY>
  <!ATTLIST ModelToGeo
```

```
    m00 CDATA #REQUIRED
    m01 CDATA #REQUIRED
    m02 CDATA #REQUIRED
    m03 CDATA #REQUIRED
    m10 CDATA #REQUIRED
    m11 CDATA #REQUIRED
    m12 CDATA #REQUIRED
    m13 CDATA #REQUIRED
    m20 CDATA #REQUIRED
    m21 CDATA #REQUIRED
    m22 CDATA #REQUIRED
    m23 CDATA #REQUIRED
    m30 CDATA #REQUIRED
    m31 CDATA #REQUIRED
    m32 CDATA #REQUIRED
    m33 CDATA #REQUIRED>

<!-- GeoToModel represents a 4x4 matrix M to convert a geo coordinate vector into a model
coordinate vector my multiplication: model = M * geo -->
<!ELEMENT GeoToModel EMPTY>
  <!ATTLIST GeoToModel
    m00 CDATA #REQUIRED
    m01 CDATA #REQUIRED
    m02 CDATA #REQUIRED
    m03 CDATA #REQUIRED
    m10 CDATA #REQUIRED
    m11 CDATA #REQUIRED
    m12 CDATA #REQUIRED
    m13 CDATA #REQUIRED
    m20 CDATA #REQUIRED
    m21 CDATA #REQUIRED
    m22 CDATA #REQUIRED
    m23 CDATA #REQUIRED
    m30 CDATA #REQUIRED
    m31 CDATA #REQUIRED
    m32 CDATA #REQUIRED
    m33 CDATA #REQUIRED>

<!-- Avatar represents the view and location of the user in the model -->
<!ELEMENT Avatar EMPTY>
  <!ATTLIST Avatar
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    z CDATA #REQUIRED
    pitch CDATA #REQUIRED
    diameter CDATA #REQUIRED
    viewingAngle CDATA #REQUIRED
    height CDATA #REQUIRED
    rotation CDATA #REQUIRED
    bluetoothID CDATA #REQUIRED
    personalID CDATA #REQUIRED
    layerPosition CDATA #REQUIRED>

<!-- a windrose to specify north direction in the model -->
<!ELEMENT WindRose EMPTY>
  <!ATTLIST WindRose
    x1 CDATA #REQUIRED
    y1 CDATA #REQUIRED
    angle CDATA #REQUIRED>

<!-- a point element represents a 3D vertex -->
<!ELEMENT Coord EMPTY>
  <!ATTLIST Coord
    id ID #REQUIRED
    ref (model) "model"
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    z CDATA #REQUIRED>

<!-- a connection element represents an edge between to vertices a and b -->
<!ELEMENT Connection EMPTY>
  <!ATTLIST Connection
    label CDATA #IMPLIED
    a IDREF #REQUIRED
    b IDREF #REQUIRED
    passability (None|Left|Right|Both) "None"
    attributes CDATA #IMPLIED
    landmark CDATA #IMPLIED>

<!-- all layers in the model ( at least one ) -->
<!ELEMENT Layer ( Image* , Object* , Item*, Display*, Box* )>
  <!ATTLIST Layer
    layerid CDATA #REQUIRED
    layername CDATA #REQUIRED
    xOrigin CDATA #REQUIRED
    yOrigin CDATA #REQUIRED
    zOrigin CDATA #REQUIRED
    width CDATA #REQUIRED
    height CDATA #REQUIRED
    style CDATA #REQUIRED
    wallHeight CDATA #REQUIRED
    doorHeight CDATA #REQUIRED
    muralHeight CDATA #REQUIRED>

<!-- the background image texture for the layer -->
<!ELEMENT Image EMPTY>
  <!ATTLIST Image
    URL CDATA #REQUIRED
    x CDATA #REQUIRED
    y CDATA #REQUIRED>

<!-- all polygons in the current layer -->
<!ELEMENT Object ( CoordIndex, Triangle+ , SubModel? ) >
  <!ATTLIST Object
    name CDATA #REQUIRED
    type (indoor|outdoor ) "outdoor"
    surfacetype
(Unknown|Building|Car_park|Footpath|Meadow|Street|Water|Elevator|Stairs|Staircase|Room)
"Unknown"
    accessability CDATA "NotPassable"
    salience CDATA "0"
    numberOfStairs CDATA #IMPLIED>

<!-- the submodel of each polygon ( optional ) -->
<!ELEMENT SubModel EMPTY>
  <!ATTLIST SubModel URL CDATA #REQUIRED>

<!-- contains a list of all Point-IDs separated by a # symbol, e.g. 1#2#3#4 -->
<!ELEMENT CoordIndex (#PCDATA)>

<!-- all polygons are additionally stored as triangles -->
<!ELEMENT Triangle EMPTY>
  <!ATTLIST Triangle
    a IDREF #REQUIRED
    b IDREF #REQUIRED
    c IDREF #REQUIRED>

<!-- an item element represents geometric primitives. An InfoPoint item is connected to a
display element. -->
<!ELEMENT Item ( Point|Sphere|Sector )>
  <!ATTLIST Item
    type                    (undefined|InfoPoint|irbeacon|rfbeacon|rfbeaconpassive|camera|
display|rfidreader|microphone|speaker|bluetooth|sound|steerableprojector) "undefined"
    name CDATA #REQUIRED
    info CDATA #IMPLIED
    display CDATA #IMPLIED>

<!-- an item that represents a 3D point geometry -->
<!ELEMENT Point EMPTY>
  <!ATTLIST Point
    ref (model) "model"
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    z CDATA #REQUIRED
    longitude CDATA #REQUIRED
    latitude CDATA #REQUIRED>

<!-- an item that represents a 3D sphere or disc geometry -->
<!ELEMENT Sphere EMPTY>
  <!ATTLIST Sphere
    ref (model ) "model"
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    z CDATA #REQUIRED
    radius CDATA #REQUIRED
    longitude CDATA #REQUIRED
    latitude CDATA #REQUIRED>

<!-- an item that represents a 3D sector geometry -->
<!ELEMENT Sector EMPTY>
  <!ATTLIST Sector
    ref (model) "model"
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    z CDATA #REQUIRED
    vx CDATA #REQUIRED
    vy CDATA #REQUIRED
    vz CDATA #REQUIRED
    angle CDATA #REQUIRED
    longitude CDATA #REQUIRED
    latitude CDATA #REQUIRED>

<!-- a Display element represents a computer display -->
<!ELEMENT Display EMPTY>
  <!ATTLIST Display
    name CDATA #REQUIRED
    server CDATA #REQUIRED
    port CDATA #REQUIRED
    password CDATA #REQUIRED
    staticURL CDATA #REQUIRED
    pointable (True|False) "False"
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    z CDATA #REQUIRED
    orientation CDATA #REQUIRED
    pitch CDATA #REQUIRED
    size CDATA #REQUIRED
    landscape (True|False) "True"
    aspect (_4to3|_16to9) "_4to3">

<!-- a box element represents a rectangular solid object. Boxes can be grouped
hierarchically: position and orientation is relative to their parent box. -->
<!ELEMENT Box (Box*)>
  <!ATTLIST Box
    name CDATA #REQUIRED
    type CDATA #REQUIRED
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    z CDATA #REQUIRED
    orientation CDATA #REQUIRED
    width CDATA #REQUIRED
    height CDATA #REQUIRED
    depth CDATA #REQUIRED
    color CDATA #REQUIRED
    textureFront CDATA #IMPLIED
    textureTop CDATA #IMPLIED>
```

C ProMan Reference Manual

The ProMan software is used by the presentation manager to present Web-based content on public displays. It runs on Windows XP/VISTA or Windows Mobile devices (PDAs) and accepts presentation requests via TCP/IP and displays Web pages using the Microsoft Internet Explorer in full-screen mode.

To install ProMan, the files proman.exe and proman-config.txt have to be copied in any folder (e.g. /Program Files) on the device. It is useful to create a shortcut in the */windows/StartUp* folder to start ProMan automatically after a reboot of the device. The configuration of ProMan is stored in the configuration file, which is explained in Table 26.

Once ProMan is started, it installs itself in the Windows system tray (Figure 144) and can be controlled by a context window that appears if the icon is clicked with the left mouse button.

ProMan will periodically send registration events to the presentation manager via the EventHeap, telling the PM that it is alive. The time between each event is set by the REGISTRATIONTIMER parameter. The event will include the IP address and port number of the device so that the presentation manager can directly connect to it via TCP/IP protocol. The event will also include the DISPLAYRESOLUTIONX and DISPLAYRESOLUTIONY values so that the PM can adapt the size of presentations to fit the display. If DISPLAYRESOLUTION parameters are given in the configuration file, they will be sent. If they are not given, or they exceed the actual screen resolution of the device, the full screen resolution of the device will be send by the event. If the user resizes the proman window, the current window size will be send by the event. The event will also include the NAME, PUBLIC, USER, MOBILE, DISPLAYSIZE, and DEVICETYPE values as defined in the config-file.

The Presentation Manager can connect to the display and send commands in plain text via a socket connection. Table 27 shows a summary of the command set that is implemented. The expected format is “command, parameter\n”.

The ProMan software is open source and can be obtained from the AI Group Wiki at

- https://w5.cs.uni-sb.de/trac/lehrstuhl/wiki/ProMan

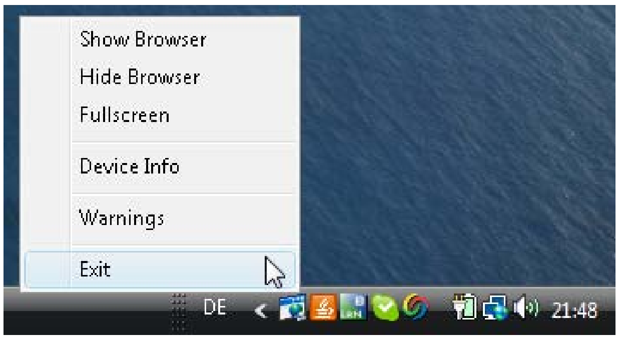


Figure 144 : ProMan icon in the system tray and context menu.

Table 26: ProMan configuration.

Parameter	Description
`SERVERPORT=6789`	Defines the port to listen for socket connections.
`EVENTPROXYSERVERNAME`	Hostname or IP-address where HTTPBridge runs. The HTTPBridge service is needed to create registration events on the Eventheap.
`EVENTPROXYSERVERPORT`	Port where HTTPBridge listens for connections.
`IPMASK=192.168.0.0`	The PocketPC might have more than one IP address. The address that is published in the registration events is selected by this mask. 0 is a wildcard. If the device's IP doesn't match the mask, warnings are given.
`REGISTRATIONTIMER=s`	Interval in seconds to send registration events.
`WARNINGS=yes|no`	If no matching IP address is found, verbose warnings are given. If unset, only warning icons will appear in taskbar.
`FULLSCREEN=yes|no`	Use the whole screen for the browser window.
`RESETTIME=hh:mm`	Time of day, when proman automatically does a soft reset. If omitted, to reset is performed.
`WINDOWPOSITION=0x0`	Position of the browser window (Desktop only).
`DISPLAYRESOLUTION=240x320`	Specify a browser window size that is smaller than the screen resolution. Only used for Desktop (Windows XP) version of Proman.
`IDLEURL=http://w5.cs.uni-sb.de`	Specifiy a Website that is displayed if the display is idle, e.g. no presentation is scheduled.
Display Information (will be sent to PM with registration events)	
`DISPLAYNAME`	Name of the display
`PUBLIC=true|false`	This is a public display
`USER=Username`	If it is not private, specifiy the owner here
`MOBILE=true|fasle`	This is a mobile display, e.g. PDA
`LOCATION=locationID`	Location where the display is situated.
`DISPLAYSIZE=n`	Screensize diameter in inches.
`FULLSCREEN=true|false`	Browser is configured in fullscreen mode.
`AUDIODEVICE=true|false`	Device has audio capabilities.
`INPUTDEVICE=true|false`	Device has input capabilities (touchscreen).
`DEVICETYPE=desktop`	Device type.

Table 27: ProMan command set and parameters.

Command, Parameter	Description
`browse, URL`	Try to load and display the webpage with the specified URL using the internet explorer
`idle`	Load and display the idle webpage as specified in the configuration file (if any).
`showwindow, WINDOWNAME`	Bring the specified window to front using the methods of the Microsoft operating system.
`hide`	Hides the ProMan window.
`createprocess, PROG.EXE`	Execute the specified program through the operating system's pCreateProcessFn() method (WinCE) bzw. ShellExecute() on Win32 systems.
`reset`	Reset the device (currently PDA only).

D Coordinate Systems

The **Cartesian coordinate system**, which is named after the French mathematician and philosopher René Descartes, defines the coordinate of each point (x, y, z) in three dimensional space as their distances from the origin of the coordinate system along three perpendicular axis, usually called x, y, and z-axis.

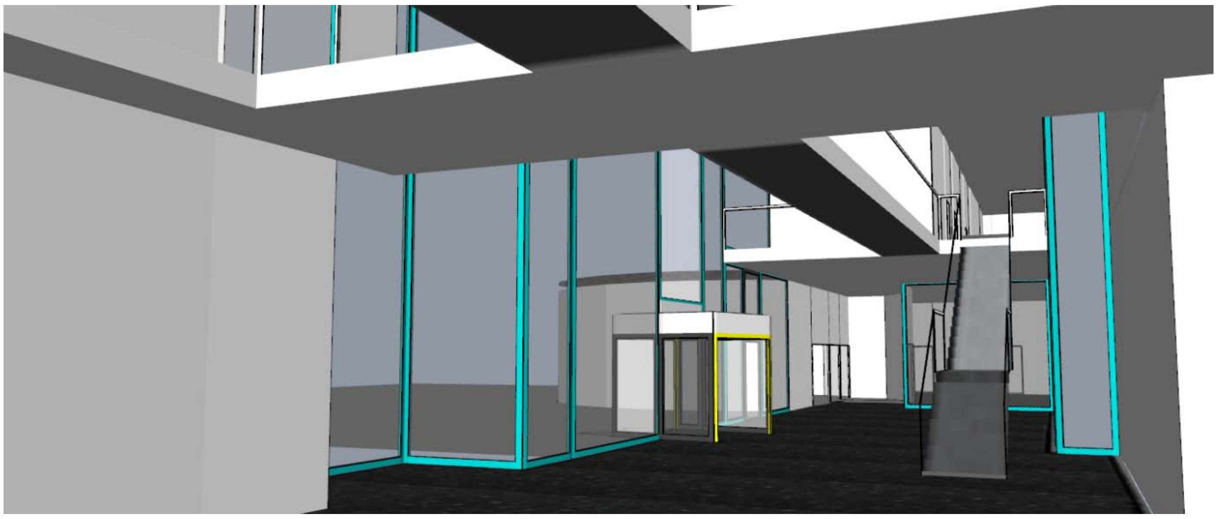
In contrast, **spherical coordinate systems** specify each point on the surface of a sphere by two **polar coordinates** *r* (the radial coordinate) and θ (the angular coordinate, polar angle, or azimuth angle, sometimes represented as ϕ or *t*). The *r* coordinate represents the radial distance from the pole, and the θ coordinate represents the counterclockwise angle from the 0° ray (sometimes called the polar axis), known as the positive x-axis on the Cartesian coordinate plane. Due to the fact that the earth's globe can be approximated to a sphere, polar coordinates are particularly useful to map locations on a global scale. A **geographic coordinate system** expresses every location on Earth by two of the three coordinates of a spherical coordinate system which is aligned with the spin axis of Earth:

- **Latitude** is the angle between any point and the equator. Lines of constant latitude are called parallels. They trace circles on the surface of Earth, with each pole being 90 degrees (north pole 90° N; south pole 90° S). The equator, an imaginary line that divides the globe into the Northern and Southern Hemispheres, is located at 0° latitude.
- **Longitude** is the angle east or west of an arbitrary point on Earth: The Royal Observatory, Greenwich (UK) is the international zero-longitude point (Longitude=0°). The antipodal meridian of Greenwich is both 180°W and 180°E. Lines of constant longitude are called meridians. The meridian passing through Greenwich is the Prime Meridian. Unlike parallels, all meridians are halves of great circles, and meridians are not parallel: they intersect at the north and south poles.
- Traditionally, degrees have been divided into minutes and seconds in a **sexagesimal** notation (e.g. 49°15'26.35"N, 7° 2'43.11"E), where each degree of longitude is sub-divided into 60 minutes, each of which divided into 60 seconds. For higher precision, the seconds can be specified with a decimal fraction. Only positive values between 0 and 90 degrees are used for Latitude, so that an additional letter ("N","S") is required to disambiguate between the northern and southern hemisphere. For Longitude, "E" or "W" represents locations to the east or west of Greenwich, respectively. Alternatively, Longitude and Latitude degrees can be represented as a signed decimal fraction (e.g. 49.257320° Lat.)

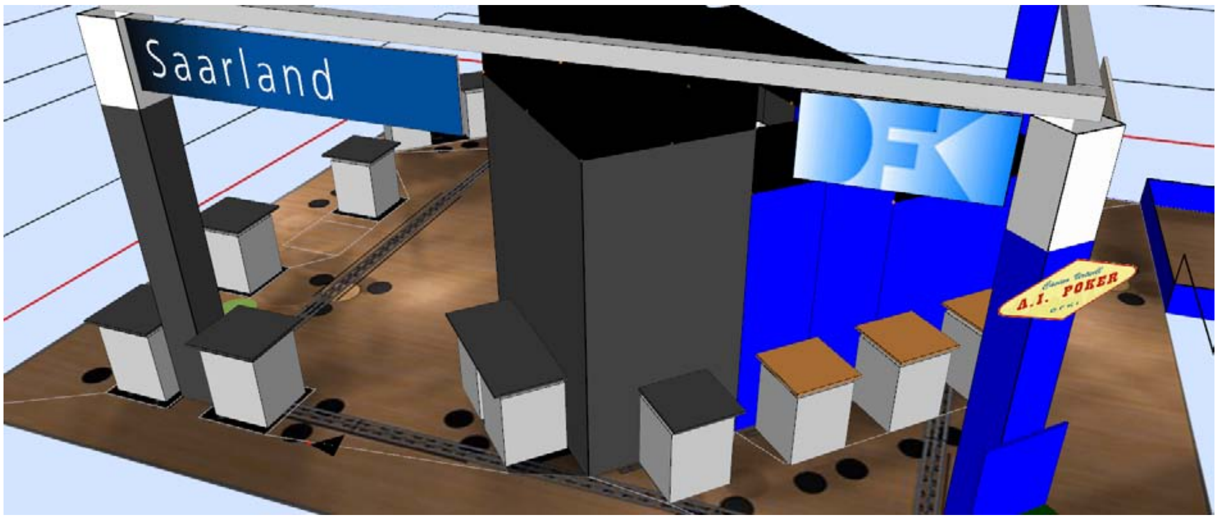
Latitude and Longitude values can use different **geodetic models** as frame of reference, which approximate the earth's geoid, which coincidences on average with the sea level, as ellipsoids. Common reference frames are the World Geodetic Model 1984 (WGS84), used by GPS, and the European Reference Terrestrial Reference System 1989[72] (ETRS89). ETRS89 is the EU-recommendation for European geodata. It is based on the same ellipsoid as the WGS84 model, but it is centered on the stable part of the Eurasian plate and diverges from the WGS84 ellipsoid by the tectonic movement of the continents.

[72] EUREF, Website: http://www.euref-iag.net/, visited 23.07.2009

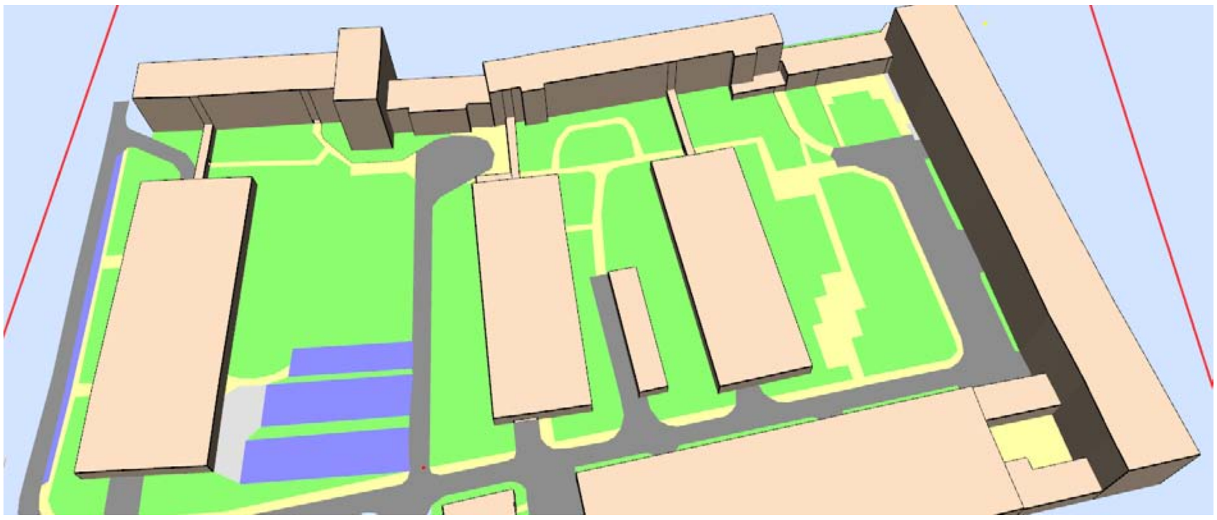
E Yamamoto Models



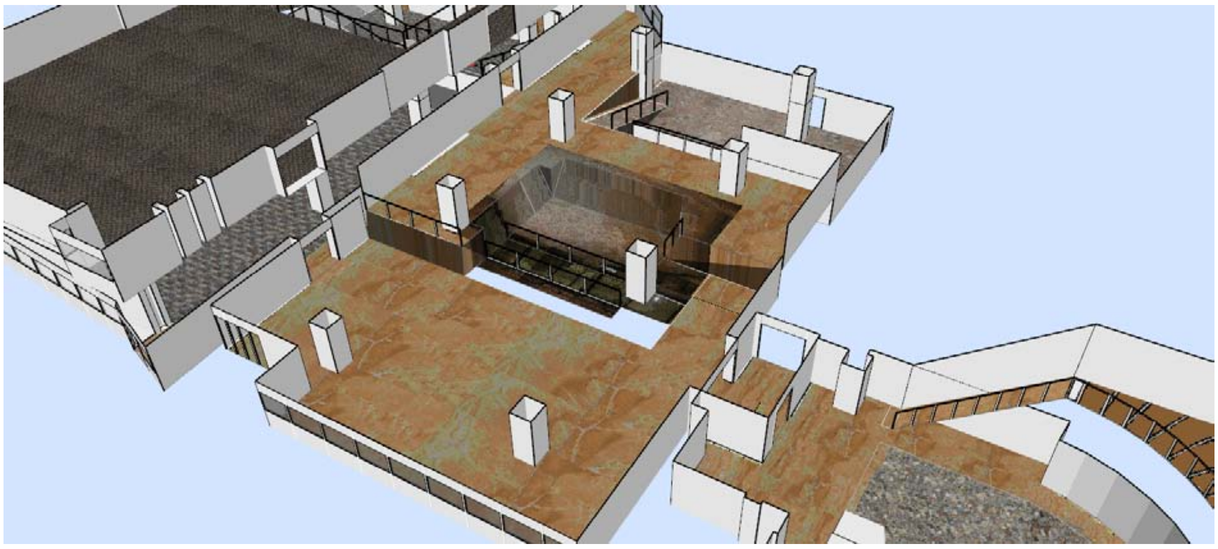
Computer Science Dept., Building E1 3, Saarland University, Saarbrücken, Germany.



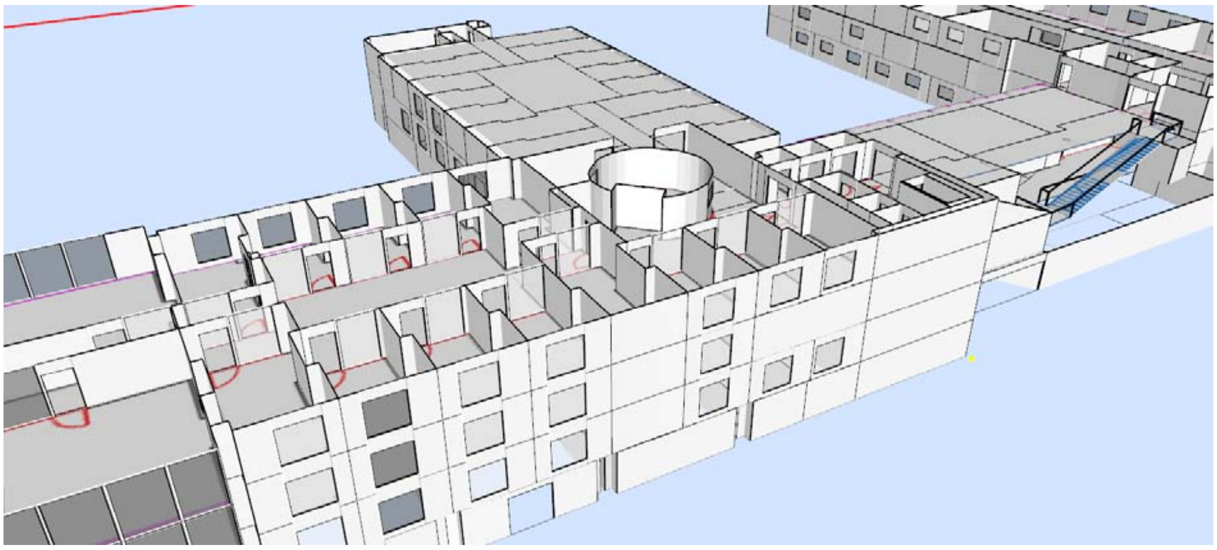
CeBit Booth, Saarland and DFKI, 2008, Hannover, Germany.



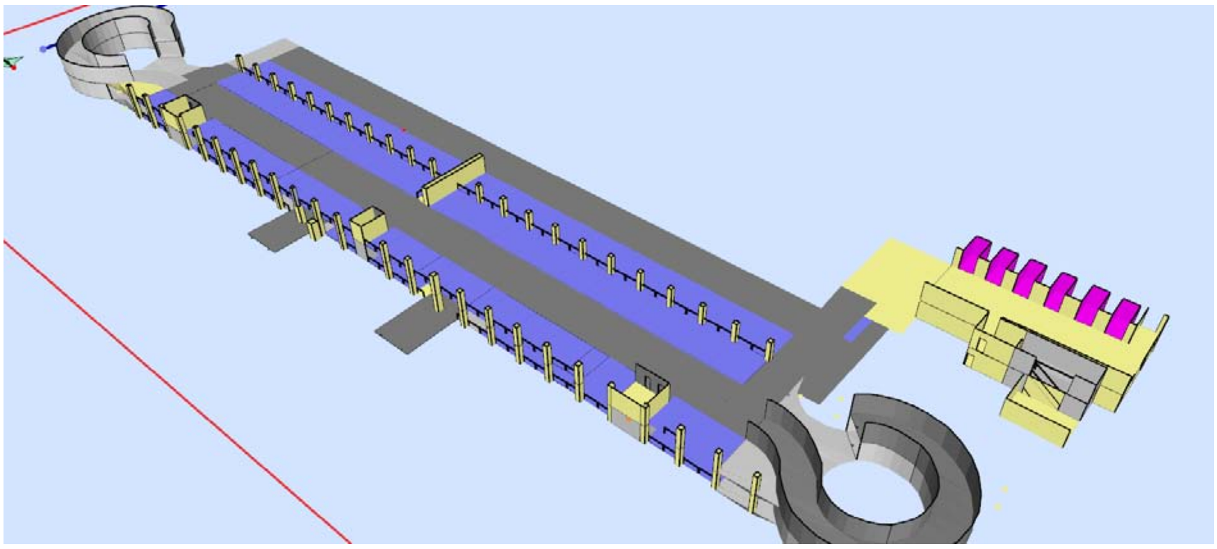
Campus, TU Chemnitz, Germany.



Hecht Museum, Haifa, Israel. Modeled by D. Hilu and R. Hilu



Infolab 21, University of Lancaster, UK. Modeled by J. Patterson



Car Park at Munich Airport, Germany. Modeled by E. Gholamsaghaee