I-centric User Interaction

vorgelegt von
Diplom-Informatiker
Stephan Steglich
aus Berlin

von der Fakulität IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuß:

Vorsitzender:	Prof. Dr.-Ing. Adam Wolisz
Berichter:	Prof. Dr. Dr. h.c. Radu Popescu-Zeletin
Berichter:	Prof. Dr. Bernd Mahr

Tag der wissenschaftlichen Aussprache: 21. November 2003

Berlin 2003
D 83

I-centric User Interaction

Stephan Steglich

Berlin 2003

Abstract

The vision of *I-centric Communications* means to take an unlimited look at human communication behavior and to adapt the activities of communication systems to it. This vision defines a user-centered approach for the realization of services and applications. It requires to start analyzing user demands to design suitable systems and services. Instead of just providing technology-focused solutions without any adaptation to individuals, an I-centric system should provide services hiding technical details and considering the individual's preferences as well as the individual's environment.

Following the vision of I-centric Communications, this thesis introduces an approach to realize *I-centric User Interaction*. This approach enhances and completes the vision by providing advanced user interaction capabilities. It answers the question whether it is possible to realize a communication system, which allows the interaction between user and services without any restriction to specific user interface technologies and in a personalized as well as ambient aware manner. Such enhanced user interaction will lead to a higher acceptance and increased usage of services.

On the one hand, the user interaction shall support different kinds of user interface technologies enabling Device Independence and ubiquitous access to the services. According to their current context and intended action, users can select the preferred and suitable way of interaction. On the other hand, the user interaction shall be adapted to the user's preferences and to the user's environment. Accordingly, this work discusses these different areas of concern, identifies necessary functions, and provides suitable solutions for each.

First, the thesis introduces and analyses the vision of I-centric Communications with special regard to the aspect of user interaction. Based on the identified requirements and areas of concern, an approach to realize I-centric User Interaction was developed. The approach, presented in this thesis, specifies a *Service Adaptation Framework* and individual models for *Personalization*, for *Ambient Awareness*, and for *Generic User Interaction* focusing on the respective areas of concern.

Finally, the thesis illustrates the results from the prototypical implementation of the presented approach, which has been pursued in several projects in parallel. These results demonstrate the applicability of the developed concepts and the fulfillment of the vision of I-centric User Interaction.

The work in the area of I-centric Communications was carried out in cooperation of the Department for Open Communication Systems (OKS) at the Technical University Berlin (TUB) and the Fraunhofer Institute FOKUS. The vision and the reference model for I-centric Communications, introduced in this thesis, are results of this cooperation.

The main research directions for the cooperation between TUB and FOKUS have been a general model for I-centric services, the service platform for I-centric services, and an approach for the interaction of users with I-centric services.

This thesis focuses on an approach for I-centric User Interaction. The general aspects of I-centric services as defined by the vision are out of scope of this thesis. Nevertheless, these aspects have been analyzed by Stefan Arbanowski, researcher at Fraunhofer FOKUS, in a second PhD thesis in parallel.

The results of this work have been contributed to different national and international projects (BMBF LiveFutura, BMBF PI-AVIda, BMBF VHE-UD, IST WSI, IST WWRI), standardization bodies (OMG, WWRF), conferences papers, and journals by introducing the vision of I-centric Communications to a larger auditorium, and by exploiting parts of the developed I-centric systems.

German Abstract

Die Vision „*I-centric Communications*“ bedeutet, einen uneingeschränkten Blick auf das menschliche Kommunikationsverhalten zu werfen, um Kommunikationssysteme entsprechend daran angepasst zu entwickeln. Diese Vision definiert einen benutzerorientierten Ansatz zur Erstellung von Diensten und Anwendungen. Dies setzt zunächst eine Analyse der Benutzeranforderungen voraus, um geeignete Systeme und Dienste zu entwerfen. Anstatt Technologie-fokussierte Lösungen ohne jegliche Anpassung an die jeweiligen Personen anzubieten, sollte ein „I-centric“ System seine Dienste ohne sichtbare technische Details und unter Berücksichtigung von Benutzerpräferenzen sowie der Benutzerumgebung darbieten.

Der Vision von „I-centric Communications“ folgend, stellt die vorliegende Arbeit einen Ansatz zur Realisierung der Idee von „*I-centric User Interaction*“ vor. Dieser Ansatz erweitert und vervollständigt die Vision durch verbesserte Benutzerinteraktionsfähigkeiten. Diese Dissertation zeigt, dass es möglich ist, Kommunikationssysteme zu realisieren, die die Interaktion zwischen Benutzern und Diensten ohne Einschränkung auf bestimmte Technologien für Benutzerschnittstellen sowie in personalisierter und umgebungsberücksichtigende Art und Weise unterstützen. Derartig verbesserte Benutzerinteraktion wird die Akzeptanz und die Benutzung von Diensten erhöhen.

Einerseits soll die Benutzerinteraktion verschiedene Arten von Technologien für Benutzerschnittstellen unterstützen, durch die die Geräte-Unabhängigkeit und der ständige Zugang zu den Diensten ermöglicht werden. Entsprechend dem aktuellem Kontext und der Absicht können die Benutzer die bevorzugte und geeignete Art der Interaktion wählen. Andererseits soll die Interaktion selbst den Benutzerpräferenzen sowie der jeweiligen Umgebung angepasst werden. Dementsprechend diskutiert die vorliegende Arbeit diese unterschiedlichen Problembereiche, identifiziert die notwendigen Funktionen und bietet entsprechende Lösungsansätze jeweils.

Die Arbeit präsentiert und analysiert zunächst die Vision „I-centric Communications“ mit Hinblick auf den Aspekt der Benutzerinteraktion. Basierend auf den identifizierten Anforderungen wurde ein Ansatz zur Realisierung von „I-centric User Interaction“ entwickelt. Dieser Ansatz, der in dieser Arbeit vorgestellt wird, spezifiziert ein „*Service Adaptation Framework*“ und einzelne Modelle für „*Generische Benutzerinteraktion*“, für „*Personalisierung*“ sowie für „*Ambient Awareness*“, die sich jeweils auf die identifizierten Problembereiche konzentrieren.

Abschließend präsentiert die vorliegende Arbeit Ergebnisse einer prototypischen Realisierung des dargelegten Ansatzes. Die Ergebnisse demonstrieren die Einsetzbarkeit der entwickelten Konzepte und die Erfüllung der Vision von „I-centric User Interaction“.

Die Forschungsarbeit im Bereich „I-centric Communications“ wurde in Kooperation zwischen dem Lehrstuhl für Offene Kommunikationssysteme (OKS) der Technischen Universität Berlin (TUB) und dem Fraunhofer Institut FOKUS durchgeführt. Die Vision sowie das Referenzmodell für „I-centric Communications“, die in der vorliegenden Arbeit vorgestellt werden, sind Ergebnisse dieser Kooperation.

Die Forschungsschwerpunkte der Kooperation zwischen TUB und FOKUS waren das „Generelle Modell für I-centric Dienste“, die „Dienstplattform für I-centric Dienste“ sowie ein Ansatz zur „Interaktion zwischen Nutzern und I-centric Dienste“. Die vorliegende Arbeit konzentriert sich auf den Ansatz „I-centric User Interaction“, der die Interaktion zwischen Nutzern und den Diensten betrachtet. Die Aspekte der I-centric Dienste werden in der vorliegenden Arbeit nicht betrachtet. Diese Aspekte wurden in einer zweiten Dissertation von Stefan Arbanowski, Fraunhofer FOKUS, analysiert und ausgearbeitet.

Die Ergebnisse dieser Arbeit wurden in verschiedenen nationalen und internationalen Forschungsprojekten (BMBF LiveFutura, BMBF PI-AVIda, BMBF VHE-UD, IST WSI, IST WWRI), Standardisierungsgremien (OMG, WWRF), Konferenzpapieren sowie Zeitschriften eingebracht, um die Vision von „I-centric Communications“ einem größeren Auditorium vorzustellen.

Acknowledgements

The ideas and concepts presented in this work have been developed in the course of several research projects, which have been undertaken at the department of Open Communication Systems (OKS) at the Technical University Berlin and the Fraunhofer Research Institute for Open Communication Systems (FOKUS).

Therefore, I express my gratitude to Professor Radu Popescu-Zeletin, my mentor, for giving me the opportunity to work in the unique combination of both institutes. He inspired this work with his strategic vision and provided continuous support, feedback, as well as motivation.

I owe Stefan Arbanowski a special debt of gratitude. Jointly, we finally managed to find the necessary time and motivation to write our ideas down. Thank you for the inspiring discussions and the constructive criticism. I will not forget those days of finalizing this work traveling around the world.

Further, I express my warm thanks to all the colleagues who participated in the related research projects and gave me necessary support and feedback. Special gratitude goes to Bernd Mrohs, Christian Räck, and Thomas Bauer, who have contributed a lot.

Additionally, I thank all the university students graduating with their work on the I-centric Portal, VHE, and SDO projects. Their contribution to the projects made it possible to write this thesis.

Of course, thanks go to all my former and current colleagues in the both institutes, providing the technical support and social environment for everyday life and research.

Last, but not least, I give my heartiest gratitude to my love Susann. Your appearance in my life and the idea of our joint future was the strongest motivation ever! I am very grateful for your understanding and patience.

Table of Content

CHAPTER 1 Introduction

This chapter describes the motivation and general objectives of this thesis giving a general introduction into the context of this work.

1.1 Motivation

During the last years, the number of commercial Internet technology based services has been grown, and the number and diversity of terminal devices, for example computers, Personal Digital Assistants, or mobile phones, has been increased. New network technologies like UMTS or WLAN offer Internet access anywhere, at any time, providing high bandwidths. These access networks have various capacities, delays, and Quality-of-Service (QoS) parameters. The numerous terminal devices have different capabilities for presenting user interfaces and for interacting with the user. Terminal devices support different types of media. A mobile phone for example may offer access to services by SMS, MMS, WAP, and voice. This heterogeneity of terminal devices and networks leads to new problems. To offer access to services through heterogeneous terminal devices and network technologies, the service providers have to adapt their services specifically for many devices and media types. This leads to high costs for development and maintenance.

Thus, today’s services do not adapt to these heterogeneous Access Mechanisms. The users are still forced to use the appropriate terminal, which might not be available or usable at any time and in any situation. Furthermore, the quality of the user interaction is still limited. The individual preferences and situations of users are not considered at all or only to an insufficient extend. These problems require suitable solutions, which should be based on the demands of users.

The scope of this thesis is to analyze and improve the interaction between user and services. Following the I-centric vision, necessary requirements and use cases are identified, which will in turn lead to a specification of concepts to enable I-centric User Interaction. The concepts describe the functions to adapt the user interaction:

- To heterogeneous terminals and communication services
- To preferences of the individual user
- To ambient information describing the user’s situation

The *Service Adaptation Framework* implementing the concept of user interface adaptation enables service developers to create services that may be accessed by users via arbitrary terminal devices and communication media. The user interface is adapted to suit the different presentation possibilities, the heterogeneous terminal devices offer. A concept to develop services independent of the specific terminal device and network technologies is necessary.

Personalization is one key factor for the success of new services offered to a broad customer base. Hence, a trend towards the application of personalization concepts in provisioning of new services can be observed. This trend is similar to the replacement of undifferentiated mass marketing approaches with the practice of market segmentation techniques in traditional commerce. Market segmentation techniques allow the subdivision of a broad and hence heterogonous customer base into several smaller and homogenous groups.

From a user's point of view, services are more attractive if they reflect personal interests and therefore decrease the so-called 'information overflow'. In addition to traditional market segmentation techniques, personalization aims at the improvement of services by providing necessary features that allow an adaptation to single users. This means, personalization allows users to suit services to fit their individual needs. This increases the quality of these services and leads to a higher degree of satisfaction. The primary goal is to attract more customers and gain their loyalty in long term.

Contextual knowledge depicts an integrative part in human-human communication. In this sense, it can be regarded as another basis for intelligent behavior of services, likewise. Whereas, explicit input and output requires attention of the user, the gathering of ambient information can extend this interaction without any explicit involvement of the user. Today, the user has to adapt to the service and cannot interact in a natural way. The knowledge of the user is in certain circumstances not sufficient to provide the explicit input. However, applications possibly depend on specific contextual information. In order to appear comfortable and easy-to-use, services should consider contextual knowledge in addition to the explicit user input.

For example, the information provided by a weather forecast service depends on time and location. The user could explicitly be requested to provide the necessary information. However, this information could be also extracted from the user's current context and the user would not be bothered with the explicit input. In this way, the user interaction would be simplified and its quality, as experienced by the user, would be increased. The goal of considering the context is that services can react in an intelligent way adapting to the user's situation in a reasonable way. Inputs can be reconstructed from the context and do not have to be explicitly inquired from the user. Based on the context information, systems can obtain the intention of the user and can react accordingly. However, in many cases this is very complex and only possible in a limited way.

1.2 Objectives and Scope

Following the I-centric vision, the goal was to develop a system that enables I-centric User Interaction, which is introduced in this thesis. This system shall support users in the communication with their services using any kind of terminal, in a personalized way, and adapted to their current context. Main features of this envisioned system are the adaptation of the user interaction to:

- The best suitable media representation according to the capabilities of the terminal and access networks
- The user's respective preferences, so that the user perceives the service in an easy-to-use and a pleasant way
- The user's current context in order to improve the user interaction

Today's human-machine interaction is still a compromise due to the existing technical restriction of user interface technologies. The human has to adapt to the particular user interface provided by a service. This also requires a learning process, which takes time and which is not accepted by all users. The implementation of the user interfaces differ from service to service, but differ even more among the different user interface technologies, e.g. speech user interfaces versus graphical user interfaces. Because of their individual nature, the different user interfaces technologies are suitable for different kinds of services and for different contexts, e.g. while driving a car, a speech user interface is more suitable than a graphical user interface.

Because of the intelligent nature of humans (who are able to interpret transmitted messages in relation to the context applying knowledge about the communication peer), human-machine interaction is still far away from reaching a similar quality like human-human interaction. The goal of I-centric User Interaction is to improve the quality of human-machine interaction by improving the intelligence of the underlying communication system. Basically, the improvement can be reached by separating the user interface from the service logic and by considering personal preferences as well as contextual information.

The objectives of this thesis are to present and to motivate an approach to realize I-centric User Interaction. This comprises a general *Service Adaptation Framework* and specialized models for *Personalization*, *ambient awareness*, and *Generic User Interaction*. The presented approach was developed based on the general vision for I-centric Communications and by analyzing and considering relevant technologies. As required, this approach reuses available and suitable technologies as well as standards to ensure openness and flexibility necessary for the practical application.

Practical implementations as well as active co-operation in various research projects (such as Eurescom P920, EU-VESPER, BMBF LifeFutura, BMBF PI-AVIda, VHE-UD, IST WSI, and IST WWRI) have backed the research on I-centric User Interaction, which leads to the concepts introduced in this thesis. In this way, the current and future trends in the area of telecommunications are reflected in the proposed approach.

1.3 Trends in Telecommunication

1.3.1 Next Generation Telecommunication Networks

At the time of writing this thesis, the third generation (3G) of wireless telecommunication networks, namely UMTS, was being deployed to the market. Characterized by increased transfer rates, this technology is expected to allow new kinds of mobile multimedia applications. Plain speech transmission is not longer the only motivator. Instead, innovative data services extending the Internet to the mobile world will provide new opportunities providing a seamless network of services to the users.

The generations following 3G, described by the term 3G beyond (3Gb), will provide a multitude of different access network technologies. Full roaming between the technology islands will be provided. The transfer rates and the capacities of individual cells will be increased continuously, by applying new technologies. First laboratory trials reached the 100 Mbit/s peak transmissions rates over the wireless link. Future terminals will comprise universal as well as specialized devices with different capabilities for realizing user interaction. In this way, the future developments will provide improved quality of service, but the heterogeneity among network and terminal technologies will remain or even increase.

This development has to be reflected by the services, which are provided to the users. On the one hand, the full potential of the new concepts should be exploited. On the other hand, the increasing heterogeneity of networks and terminals has to be considered, which requires an adaptive behavior of the system.

1.3.2 Information at any Time, any Place, and in any Form

The vision of information at any time, at any place, and in any form has been investigated by recent research activities. In last years, also the market has adapted that slogan to promote its offers. It has already been shown [PFEIFER1], that it is feasible to deliver any kind of information to any kind of terminal. To enable such applications, conversion processes have to be integrated in the communication systems. These conversion processes can be used for the automatic service delivery and for the remote access to stored messages. In both cases, the conversion processes adapt a certain kind of message to the terminal the customer uses.

This approach tackled primarily the heterogeneity of terminals and communication services and provided first solution concepts, such as unified messaging approaches. Today, there is a number of according products available. However, these approaches are limited to telecommunication services enabling a more comfortable and flexible user-user interaction as described in the following section.

1.3.3 Unified Messaging Approaches

The concept of *unified messaging* has emerged from the research of Personal Communication Systems (PCS). Having quickly reached industrial relevance [NANNEMAN, DECUMS, VDMEER, ARBMEER1, ARBMEER2], it addresses the task of overcoming the multiple mailbox approach of today's telecommunication scenarios, with separated facilities for e-mail, voice messages, fax reception, etc. This coincides with the vision for future intelligent communication, to deliver 'information any time, any place, in any form', as described in the Virtual Home Environment (VHE) concept [ETSITS22.70] within the emerging Universal Mobile Telecommunication System (UMTS) standards [ETSITS22.01].

Global reachability has been enhanced today by mobile equipment, so now the users want to manage and to control this accessibility in order to maximize or to filter it – independent of their location, the used communication service medium, and the applied human communication interaction (asynchronous or synchronous). Additionally, they want to have access to their asynchronous message store from everywhere and in any form, i.e. using the available equipment. When accessing their received messages remotely, the users want to focus on the most considerable aspects and postpone or discard unimportant material.

This approach is applied to the concept of Personal Communication Support (PCS) [ECKARDT, GUNTERMANN], which provides people with a new dimension in communication (Figure 1). In general, the concept allows users to establish their own personalized communication environment by addressing three important aspects, namely:

- *Personal Mobility*, which denotes the mobility of the user in fixed networks and wireless networks, allowing the user to make use of communication capabilities available at different locations, i.e. at any place and any time; including the automatic determination of their current vicinity, i.e. location awareness.
- *Service Personalization*, including personalized call / reachability management, which allows the user to configure the communication environment and control the reachability according to the specific individual needs, i.e. if, when, where, for whom he will be reachable, possibly supported by concepts of content screening.
- *Service Interoperability* in distributed multimedia environments between different types of communication services and terminals, allowing users to maximize their reachability. In this context, multimedia capabilities are required that enable dynamic, intelligent content handling and conversion of different media types and media formats in order to deliver information in any form.

A TMN-based *Personal Communication Support System (PCSS)* [ECKARDT1, MAGEDANZ] had addressed the first two aspects of PCS, namely personal mobility and service personalization, in a project at GMD FOKUS / TU Berlin, OKS. In order to provide full PCS capabilities within emerging CORBA/TINA based environments, both departments started in 1996 the joint development of a *Personal Communications Support in TINA* [ECKARDT2] and an *Intelligent Personal Communication Support System (iPCSS)* [PFEIFER].

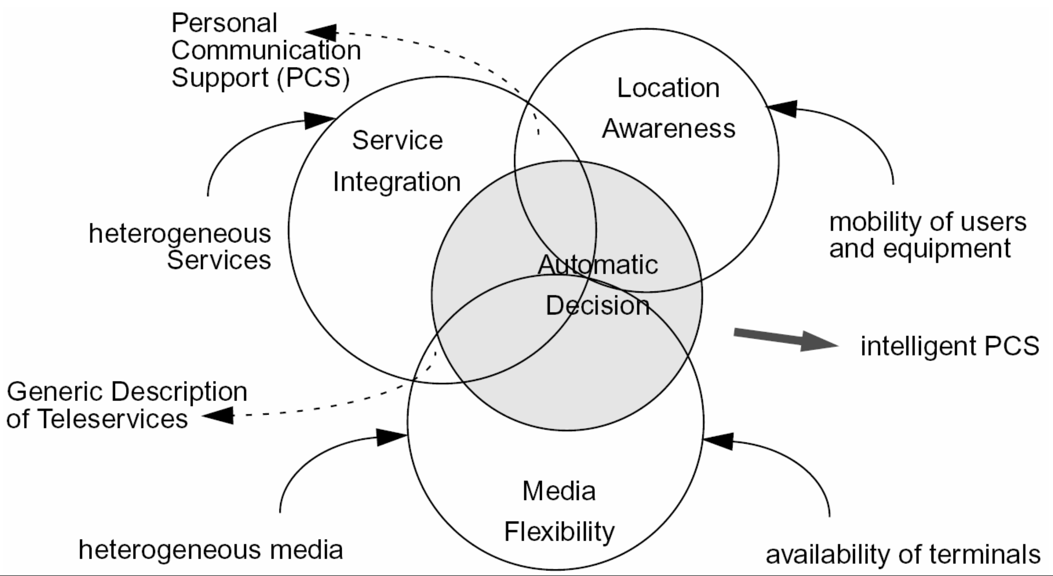


Figure 1-1: Enterprise View of intelligent Personal Communication Support

The term ‘intelligence’ refers to the capability of the system to make certain decisions within user defined limits by itself, therefore relieving the user from pre-planning every possible situation in the communication environment.

1.3.4 The Virtual Home Environment

The Third Generation Partnership Project (3GPP) technical specification [3GPP-TS22.121] discusses the Virtual Home Environment (VHE) concept as it is seen by the 3GPP. According to this specification, the idea behind the VHE concept is to provide service providers with a *Personal Service Environment (PSE)* that is portable across network boundaries and between terminals. The PSE presents users the same Personalization features, user interface customization, and services regardless of the underlying network, used terminals, and their current location. The so-called Home Environment (HE) provides a PSE, which is portable only between terminals. When roaming across different networks, the VHE enables users to take virtually their personal home environment, which includes their preferences and subscribed services, with them.

The Global System for Mobile Communication Memorandum of Understanding (GSM MoU) [GMPCSMOU] plenary initially introduced the term VHE in 1995. It was proposed as the solution for the evolution from several different second-generation mobile systems to a single world standard for IMT-2000. Thereupon several standardization bodies, research facilities, and organizations worked on the VHE concept. The aim is to create a flexible platform that allows users to have seamless access to their subscribed services. Seamless access means that all services have a consistent look and feel, even when users roam across different networks or use different kinds of terminals.

Due to the multitude of participating groups and organizations, there are many different definitions of the term VHE. In the following, a selection of exemplary definitions is presented in order to give an overview about interpretation possibilities of the term VHE:

“VHE means that the user will have the same interface and service environment regardless of location.” [UMTSR1]

“VHE is a system concept for service portability in the Third Generation across network borders.” [GMPCSMOU]

"VHE provides operator specific services that are accessible by the user even when this user is roaming outside the Home network. The establishment of this concept realizes that service provision and network operation may be separated, allowing services to the offered by networks other than those providing the home or visited call processing capabilities." [ITU-Q1711]

"VHE is defined as a concept for Personal Service Environment (PSE) portability across network boundaries and between terminals. The concept of the VHE is such that users are consistently presented with the same personalized features, user interface customization and services in whatever network and whatever terminal (within the capabilities of the terminal and the network), wherever the user may be located." [3GPP-TS22.121]

Some fundamental aspects deriving from the VHE concept are service personalization, service scalability, and service portability. Based on these basic characteristics, a user will be able to access to a wide set of services. The term VHE does not stand for a single component or protocol; it depicts rather a vision independent from technical solutions and describes advanced business scenarios and models. According to [UMTSR1] it is regarded as a key concept within the Universal Mobile Telecommunications Service (UMTS) and networks of the following generation, i.e. beyond 3G.

The technical specification [3GPP-TS22.121] lists some top-level criteria for the provisioning of an eligible VHE. In this context, the VHE consists at least of:

- A consistent set of personalized services
- A customizable user interface (within the capabilities of the used terminals)

From the perspective of the users, the services are independent from the underlying access networks (e.g. fixed, mobile, or wireless networks) and globally available when roaming across different networks.

The VHE concept raised the need for standardizing several requirements such as the interoperability between different network technologies, the interworking between different layers, and the secure exchange of user information. However, the key concept of VHE (i.e. services) cannot be standardized due to the competitive environment that network operators and service providers face. It is expected that only a minimum set of services and their respective specifications will be agreed among a number of network operators and service providers. It is however necessary to define and specify the means for service creation, service operation, service management and service portability [ETSITS22.01, GMPCSMOU].

The limitations and the drawbacks of the current mobile terminals and services offered by the networks is the main motivation that led to the development of new technologies and services that will be offered to the users independently of where they are using the mobile terminal. In this way, users will be able to have advanced services everywhere they are traveling independently from network, access, or terminal. A new era of competition will be opened between different networks and service providers. Traditional methods of introducing a new service are slow and can hinder operators working in today's fiercely competitive environment.

The VHE vision was subject of a number of international research projects in the telecommunication and the information technology domain. These research projects have provided first approaches and concepts, which were realized and demonstrated by prototype implementations. However, not all necessary aspects are solved today completely and the developed concepts are still discussed in the various relevant standardization bodies. Because of this, the complete VHE concept is not part of the already deployed and upcoming 3G networks, but some basic ideas are already adopted and supported by the vendors of telecommunication equipment. However, the principles of the vision are partially adopted in newer research visions, such as the user-centered design and development as described by the vision of I-centric Communications, the starting point of this thesis. In this way, these newer visions will continue determining the goals and objectives of ongoing and future research activities.

1.3.5 Adaptive Telecommunication Services

Telecommunication systems are increasingly improved by extending them with intelligent adaptation capabilities. In order to improve the quality of service and hence the acceptance by the users, services are made more intelligent, which means that they adapt to the specific requirements and situation of the user. The adaptation capability is a key feature and a very natural phenomenon in human-human communication and is still a drawback in human-machine communication.

For a successful adaptation, as much as possible information about the user is required. This information comprises user preferences and the user's current situation, which can be described by the term *context*. A context contains static data but also dynamic data according to the user's current situation. Beside the explicit input of the user to the service, the contextual information can be considered additionally in order to improve the service usage. This leads to a new, smarter, and in turn more user-friendly representation and behavior of the services.

The following sections give some brief overview of the current relevant trends in the telecommunication domain.

1.3.5.1 Personalization of Telecommunication Services

The term *personalization* in the scope of computer science is used when talking about applications, which can be configured to consider individual preferences, e.g. the ring tone and background picture of mobile phones. In last years, the individual shaping of services and devices became to a very successful money-spinner, e.g. the colorful and fashionable mobile phones covers. [KOBSA] defines a personalized application as 'a hypermedia system which adapts the content, structure and/or presentation of the networked hypermedia objects to each individual user's characteristics, usage behavior and/or usage environment.'

The general understanding of personalization can be defined as follows:

The *personalization* of a service means to consider and adapt to the user's preferences in the provisioning of the service. This leads to a service appearance and behavior specific to the user.

In order to be able to enable service providers to adapt their services to their customer's individual needs, the necessary personalization data have to be gathered first. These data establish the common ground for the subsequent adaptation decisions to be made. First personalization concepts are already gaining momentum in various business domains, because of the improved service quality, which the users perceive, and the strong competition among service providers to attract customers.

For example, [KOBSA] states: 'Until recently, electronic commerce on the WWW was mass-oriented. Business offers and services were uniform and geared to suit the largest possible buyership. As is the case in traditional commerce, one can, however, expect that in the long run those companies will survive more easily that can establish, maintain and extend a customer base by delivering tailored products and services and thereby create stable long-term relationships with repeat customers.'

This means that the personalization feature will be an essential characteristic of future telecommunication services. Today, there are different realizations of the personalization features, which are specific to the respective applications and which are based on diverse concepts. The different realizations do not cooperate and do not share the underlying personalization data resulting into a different representation to the users. Because of the different approaches and the resulting impacts to the users, the extent of applying personalization concepts is still limited and hindered. Hence, there are many research activities in this area working on unified and interoperable personalization concepts.

Basically, all personalization concepts base on the gathering and interpreting of the personalization data. These can be divided into two different classes:

- General user data
- Service specific usage data

General user data denote information about personal characteristics of the user. This information can be divided into several categories. These categories include demographic data, user knowledge, user skills and capabilities, user interests, goals and plans. Some information may overlap with or can be refined by the usage data. These user data categories have been the basis for personalization in a large number of systems developed so far. Especially demographic data and user interests and preferences are the basis for many of today's personalized websites.

The **service specific usage data** are related to the user's interaction behavior and to the user's service usage. This means, they comprise observable usage regularities. These data can often be observed by the system directly, whereas the general user data could require one or more additional acquisition steps.

The acquisition of general user data can include the interrogation of the user. For example, demographic data can be provided by the user only. However, there are several possibilities how this user data provisioning can be realized. Some initial questions could be posed during the system registration process. This method is often used in modern personalized websites. In addition to that, subsequent interviews can be used. However, as described in [KOBSA] the 'self-assessment is error-prone since users are often not correctly aware of things like their own capabilities.'

Automatic learning methods could be applied additionally or alternatively for gathering user data. However, [MANBER] states: 'A major weakness of such systems is their unpredictability. Most users expect to have at least an intuitive notion of what is given to them, and they expect to see the same behavior consistently. Being surprised is wonderful if it is entirely a positive surprise, but overall, being unpredictable is a negative. In particular, if people are not sure, how things work, they are less prone to experimentation, because they are afraid of breaking something, or that they cannot restore the previous state. Any personalization feature should encourage experimentation.'

1.3.5.2 Context-aware Telecommunication Services

In addition to personalization data, the context, in which a service is used, can also be considered as a relevant source for the reasonable adaptation of services. The context describes ambient information consisting among others of information about the software environment, hardware environment, and locale information. Human imply contextual knowledge! If a person says: '*If you go out, take an umbrella with you!*', it is likely that it is raining or at least that it is going to rain. Human can easily grasp the context and react accordingly, which leads to an intelligent and capable communication.

Why consider the context in the service provisioning? As seen above, contextual knowledge is the basis for intelligent behavior and integrative part in human-human communication. Position data are one famous example of contextual information. Applications considering position data are regarded as being *location-aware*. However, location is only one sphere of the whole context. It depends on the capabilities of a certain system to sense contextual information and also on the requirements of the target application, which contextual information can be considered.

Importance of Context in Everyday Life

The following excerpt is taken from Lewis Carroll's 'Through the Looking Glass'[1]:

> "*[…] that shows that there are three hundred and sixty-four days when you might get un-birthday presents –* "
> "*Certainly,*" said Alice.
> "*And only one for birthday presents, you know. There's glory for you!*"
> "*I don't know what you mean by 'glory,'*" Alice said.
> Humpty Dumpty smiled contemptuously. *"Of course you don't - till I tell you. I meant 'there's a nice knock-down argument for you!'*"
> "*But 'glory' doesn't mean 'a nice knock-down argument,'*" Alice objected.
> "*When I use a word,*" Humpty Dumpty said, in a rather scornful tone, "*it means just what I choose it to mean - nothing more nor less.*"
> "*The question is,*" said Alice, "*whether you can make words mean so many different things.*"

The above excerpt shows a specific use of the English language in which the meaning of particular words is determined by the speaker. Moreover, there are some words or phrases, which are ambiguous. They are subject to the vagaries of the natural language they are expressed in or their meaning depends on the current situation of the involved persons. Consider the following example:

"*Please remove all your clothes when the light goes out!*" says a sign on a washing machine in a public laundry. The citation alone can be understood in different ways. However, taking the supplied information about the current situation into account it becomes clear that the washing machine should be emptied once the cleaning process finished.

Naturally, any information about the current situation is automatically applied before taking any action. This is done in order to act accordingly. Another example is common sense reasoning. It is normally based on some background information which is often nothing else than situation dependent information. This situation dependent information includes miscellaneous parameters from the current environment. In the above example some eligible parameters are:

- The related object (the status indicator on the washing machine)
- The location (a public laundry)
- The action (removing all clothes from the washing machine), to which the phrase refers

This situation dependent information is called context. A more detailed definition is given in the next section. In this particular example, the supplied context indicates that the sign does not request to undress oneself in the case of a shortfall of the laundry's lighting.

Definition of Context

There are many different definitions of the term context. Often they suit only their specific application domain. Rakotonirainy offers in [RAKOTONI] a more general definition that applies to the scope of this work:

Context (from an entity's viewpoint) is information that can be used to characterize the situation of an entity and can be obtained by the entity, where an entity can be a person, place, physical or computational object.

[1] [CARROLL]

A more specific definition of the term context in the field of computer-human interaction is given by Dey in [DEY].

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.

Dey closely associates the current context with the interaction between a user and an application. The provided information has to be useful in order to fulfill a certain task. While Rakotonirainy offers a general definition of the term context, Dey's definition proves to be more useful in the context (in Rakotonirainy's sense) of this work.

Following Dey's approach it has to be defined, what information seems to be relevant and what information can be omitted. The following section discusses this topic.

Context Types

Dey introduces four types of relevant context information in order to categorize the gathered information. This is done to facilitate a structured and organized processing methodology. According to [DEY] there are four primary context types:

- Identity
- Location
- Activity
- Time

On the one hand, the primary context types can be used to answer elementary questions beginning with the interrogative pronouns 'who', 'where', 'what', and 'when' when speaking about the situation of a particular entity.

On the other hand, the primary context types can also be used as indices to other context resources, which further deepen the elementary information contained in the primary context. Given a person's identity, many pieces of related information can also be gathered, e.g. the phone number, e-mail address, and the date of birth. The location information can be used to determine other objects or persons nearby.

According to [DEY], the information contained in a primary context can be used as an index to obtain secondary contexts (e.g. the date of birth) for a particular entity as well as different primary contexts for related entities (e.g. in the same location). The term secondary context stands for all information that can be derived from one primary context type or from a combination of the four primary context types contained in a particular context (e.g. a local weather forecast depends on location and time data in order to provide any useful information).

What is Context-aware Computing?

Schilit and Theimer introduced the term context-aware computing back in 1994 [SCHILIT]. They considered software that "adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects overtime" as characteristics of context-aware computing.

In contrast to Schilit and Theimer, Dey gives a more general definition of the term context-aware computing in his later work [DEY]:

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.

There is no exact definition which features are at least required to call an application context-aware. In order to name some of the most important features of context-aware applications that have been proposed in miscellaneous studies, it can be said that context-aware applications aim at providing intelligent, customizable, situation-dependent, and self-adapting services for application domains such as personal communications, ubiquitous computing, etc.

1.4 Organization of this thesis

Following this general introduction into the subject of this thesis, **chapter 2** introduces the vision of I-centric Communications. This vision is the foundation of this work specifying a reference model, the necessary terms, and different viewpoints. Accordingly, it provides the motivation and the high-level objectives. After the general presentation of the vision, this chapter introduces more detailed the aspect of I-centric User Interaction, which is an integrative part of the general vision and represents the subject of this work. Accordingly, general requirements and viewpoints are discussed initially, which are later refined and taken up in chapter 3.

Chapter 3 introduces an according approach to realize I-centric User Interaction. This approach consists of a Service Adaptation Framework and three suitable models for Personalization, for Ambient Awareness, and for Generic User Interaction respectively. After the general introduction into the solution approach, a Service Adaptation Framework is introduced. This depicts an architectural framework implementing the individual function of the solution models. The necessary functional elements are identified and their interactions are described. This results in a platform independent specification, which depicts the foundation for the prototypical realization. The specification of the framework is based on three individual models, each for a relevant area of concern, i.e. for Personalization, Ambient Awareness, and Generic User Interaction. Whereas the specification of the Service Adaptation Framework already introduces the main concepts of each model briefly, the individual models are presented detailed in separated sections after the introduction of the Service Adaptation Framework.

Chapter 4 describes a prototypical realization of a system realizing I-centric User Interaction. This system, denoted as the I-centric User Interaction portal, implements the Service Adaptation Framework as introduced in chapter 3 in order to support the device independent, personalized, and ambient aware appearance of the services. Furthermore, exemplary services, which have been realized on the portal platform, are presented in order to demonstrate the capabilities and the potential of the I-centric User Interaction approach.

Chapter 5 provides the summary and the outlook on work already in progress and possible in future. References of related literature as well as a list of acronyms are given in **chapter 6**. An appendix, **chapter 7**, containing some technical details and examples closes this thesis.

CHAPTER 2 I-centric Communications

Following the vision of Ambient Intelligence, the department for Open Communication Systems at the Technical University Berlin has coined the term I-centric Communications, which picks up the idea of providing services based on the individual communication behavior of human being. This chapter first introduces the vision of I-centric Communications including a reference model, which has been used to develop and to design the I-centric User Interaction approach. This approach focuses on the aspect of the interaction of an individual with objects in the individual's communication space. This chapter analyzes and discusses this aspect in detail in order to derive the necessary requirements on an I-centric User Interaction approach.

The work in the area of I-centric Communications was carried out in cooperation of the Department for Open Communication Systems (OKS) at the Technical University Berlin (TUB) and the Fraunhofer Institute FOKUS. The vision and the reference model, introduced in this chapter, are the results of this cooperation. The results have already been contributed to several national and international projects, conference papers, journals, and diploma theses.

The main research directions for this cooperation have been:

- *A general model for I-centric services*
- *The service platform for I-centric services*
- *An approach for the interaction of users with I-centric services*

This thesis focuses on an approach for I-centric User Interaction. The general aspects of I-centric services as defined by the vision are out of scope of this thesis. Nevertheless, these aspects have been analyzed by Stefan Arbanowski, researcher at Fraunhofer FOKUS, in a second PhD thesis[2] *in parallel. The following sections (section 2.1-2.4) represent the basic framework for both dissertations. They have been developed together and therefore are part of both PhD theses.*

In this sense, both PhD theses are complementary. They are based on the same vision, focusing on different aspects of I-centric Communications. Section 2.5 introduces the basis of this PhD thesis. It relates the research tasks, which are unique for this thesis, to the vision of I-centric Communications. Chapter 3 proposes a Service Adaptation Framework including different models for the realization of I-centric User Interaction. In chapter 4, an implementation of the introduced concept is described. It represents a first prototype of a system that provides I-centric User Interaction. This system was developed at TUB. Due to the cooperation with Fraunhofer FOKUS, first steps towards an integration of both works, the I-centric User Interaction and the I-centric service platform, have been carried out together.

[2] Arbanowski, St.: 'I-centric Communications', PhD thesis, Technical University Berlin, Falkultät IV, 2003

2.1 Vision

The communication behavior of human beings is characterized by frequent interactions with a set of *objects* in their environment. Humans solve the problems of their daily life, e.g. money and bank accounts need to be managed, food has to be bought and to be prepared for eating, movies are watched for entertainment, places are visited and news are consumed to improve education, and other people are met for discussions. The set of objects, controlled by each individual human, define its *individual communication space* as shown in Figure 2-1.

A communication space of an individual is limited: 'I do not know everybody in the world, I am not interested in everything, and I do not have all necessary devices required by all communication services everywhere at all times'.

Furthermore, individuals are interested semantic and not necessarily in the kind of presentation of a specific service. Services in an individual communication space have to support the quality of the human senses, and since quality of senses is individual, they have to *adapt* their presentation to each individual automatically. Services have to adapt to the life stage and the environment of each individual.

In the former days, the communication space of human beings has been limited to the actual surrounding physical environment (like a home or an office) because of the spatial range of human senses. With the introduction of telephony, the range was expanded. It became possible to talk with people regardless their location. With asynchronous services like email and Short Message Service (SMS), the dimension of time expanded. People can send emails and do not need to care, whether the addressee is ready to receive the message or not. That is, technology has eliminated distances in time and space or at least made these boundaries almost unperceivable. By this means, today's communication services act as a prolongation of human senses and extend the individual communication space.

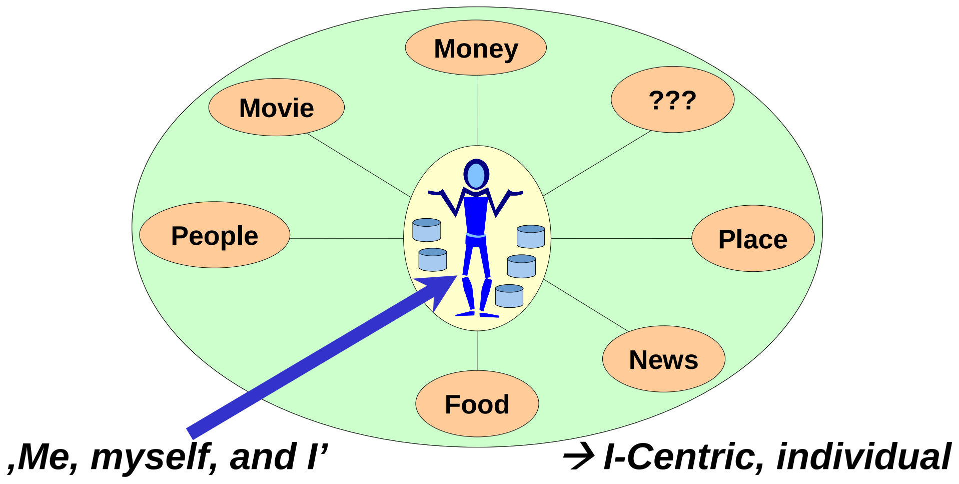


Figure 2-1: Individual communication space

Following this view, a new approach is to build communication systems not based on specific technologies, but on the analysis of the *individual communication space*. The result is a communication system that adapts to the specific demands of each individual (*I-centric*). Such a communication system will act on behalf of human's demands, reflecting recent actions to enable profiling, and self-adaptation. I-centric services adapt to individual communication spaces and situations. In this context 'I' means I, or individual, 'Centric' means adaptable to I demands and a certain environment. [ICS1]

The rationales above require intelligence in service provisioning in order to personalize, adapt to situational and environmental conditions, to monitor and to control the individual communication space. An I-centric communication system will provide the intelligence, which is required for modeling the individual communication space of each individual by adapting to its interests, environment, and preferences.

The multitude of devices, wearables, different telecommunication technologies, positioning and sensing systems are considered as enabling technologies for I-centric Communications. Universal information access (including service interworking, media conversion), flexible management of equipment and facilities (e.g. smart homes [WELLNER]), and personal communications (supporting personal mobility and terminal mobility [ECKARDT1]) form the basis of such systems.

2.2 I-centric Communications – Basic Terminology

I-centric means to take a bottomless look at human behavior and to adapt the activities of communication systems to it. Human beings do not want to employ technology. They rather want to communicate acting in their individual *communication space*. They meet with others to talk, to celebrate, they read and travel, they are listening to news or to music, they take decisions, etc. This abstract description of humans' communication activities requires a set of definitions that allow the mapping of abstract requirements (or wishes) to the physical communication environment later on. In the following, the basic terminology to describe I-centric Communications systems is introduced.

As discussed above, the human communication behavior is characterized by frequent interaction with a set of other humans, information sources, and devices within or outside humans' vicinity. All entities, humans are interacting with, will be called *objects* further on. They can be activated or deactivated[3] by an individual, or environmental conditions, to perform an action according to specific needs of an individual. Objects are directly addressable, and can represent one or more physical entities performing a certain service.

Object: An object is a logical representation of hardware or software entity, or even a representation of a certain individual, and provides well-defined services from the perspective of an (other) individual.

To model the interaction of human beings with objects of their individual communication space a general model applicable for all kinds of objects is needed. Objects will be used differently in different contexts by different individuals. In addition, a mechanism is needed for managing the use of objects (e.g. monitoring the activities of each object is used to profile, or to account service provisioning). The focus regarding individual-object interaction is to provide services of objects in a generic way.

To enable ad-hoc interaction of beforehand unrelated objects, an interaction model between objects is needed. This will allow the dynamic collaboration of objects for a specific purpose. Together with an organizational model, which describes relations between objects like ownership issues, such kind of interaction can be used to stimulate social behavior of objects, like multi-agent systems [ARB] do, to perform a specific task.

The basic idea of objects, to model real-world entities, has to be reflected by the I-centric service architecture. General procedures for wrapping legacy technology have to be developed facing the fact that environmental constraints can affect the design of a distributed system [ODP, X.901-904]. Middleware concepts have to be selected that on one hand hide the hetero-

[3] Activation or deactivation in this sense is an abstraction of using an object. The 'usage' can refer to a complex task involving multimodal user interaction or complex business / service logic inside or on the outsite of an object.

geneity of physical resources (represented by objects of individual communication spaces), but on the other hand support vertical integration, if necessary.

A general methodology how to select/design objects of individual communication spaces has to be developed. An I-centric system should support massive numbers of objects and should be tolerant against object failures. The population of objects is always changing because they spontaneously enter/leave/roam the environment and hence the communication space. Already standardized mechanisms for naming, lifecycle, monitoring, fault tolerance etc. have to be taken into account to determine whether they suit the requirements of I-centric Communications.

Due to changes in human being's daily live, the amount or the concrete instances of objects are changing over time. Nevertheless, the sum of objects, with which an individual might interact, forms his *individual communication space*. Objects may pertain to different communication spaces. They can be controlled by individuals, other objects, or services. Individuals can directly ask for a service to be performed by an object, whereas environmental condition may influence the status of objects indirectly. Communication between different individuals takes place by sharing objects of their communication spaces. In this case, objects representing communication facilities in the different communication spaces will be connected to establish a physical connection between two individuals. What kind of physical resources are used for the communication is decided dynamically and depends on individual preferences of involved parties, their available communication facilities, and additional ambient information. The process of how to select and activate objects and physical resources underneath is one of the main activities of an I-centric communication system and will be introduced in section 2.2.1.

Individual communication spaces are growing and shrinking in the time axes based on the individual life stage, personal interests, working and living environments, and the availability of new kinds of telecommunication services and devices. Meeting new friends or entering a room, which provides certain telecommunication devices, enlarges the communication space, whereas leaving the same room will reduce the size of the communication space dynamically. This process of growing/shrinking the communication space has to be supported by an I-centric communication system in two ways. On one hand, the system must perform this process automatically, but on the other hand, the individual must have the possibility to include and exclude objects explicitly.

<u>Individual Communication Space:</u> The individual communication space of a certain individual is defined by a set of objects this individual might want to interact with. The size of the individual communication space varies over time due to the appearance or disappearance of objects.

Each individual has only *one* individual communication space. It contains all objects this individual might want to perform requests on. Objects that pertain to individual communication spaces of different individuals must handle concurrent access from different individuals or must delegate the concurrency control to the I-centric communication system.

Although the individual communication space provides a repository of objects for an individual there is no partitioning of objects in the communication space itself. The concept of context, personalization, and ambient-awareness provide necessary mechanisms to structure, classify, group, and even to partition objects by several means, i.e. relations between objects to define dependencies, or relations between objects and individuals to define ownership issues or usage rights.

2.2.1 Context and Active Context

An individual communication space provides a set of objects, whose services an individual can use, to achieve its goals. In addition to that, the concept of the *context* provides the definition of relationships and causalities between different objects of an individual communication space and the individual. A context represents a 'universe of discourse' in an individual communication space. Individuals communicate with objects in their environment in a certain context.

A context can be seen as a metaphor for:

- Which objects of an individual communication space, an individual wants to use in parallel
- Which objects have to be taken into account to fulfill a certain wish of the individual
- Whether and how objects are affected by activities of other objects

Objects may pertain to different contexts (even to contexts of different individuals), because individuals might want to have a certain object involved in different activities.

Context: A context represents a certain 'universe of discourse'. It defines relationships and causalities of an individual to and between particular numbers of objects of its communication space.

Contexts are independent from any concrete environment. If an individual wants to *act in a certain context* this context has to be activated. An *active context* defines the relationship of an individual to and between particular numbers of physical resources (see Figure 2-2) at a certain moment in time, in a certain environment. The activation and deactivation of a context should usually be done automatically just by analyzing individuals' activities, but an individual should also have the possibility to do this explicitly.

Activating a context means:

- The identification of objects that are required by the context
- The evaluation of the relationships and causalities between objects defined by the context
- The discovery of the actual vicinity of the individual to identify physical resources that provide the functionality required by the identified objects
- The activation/configuration of these physical resources to perform the task required by the context

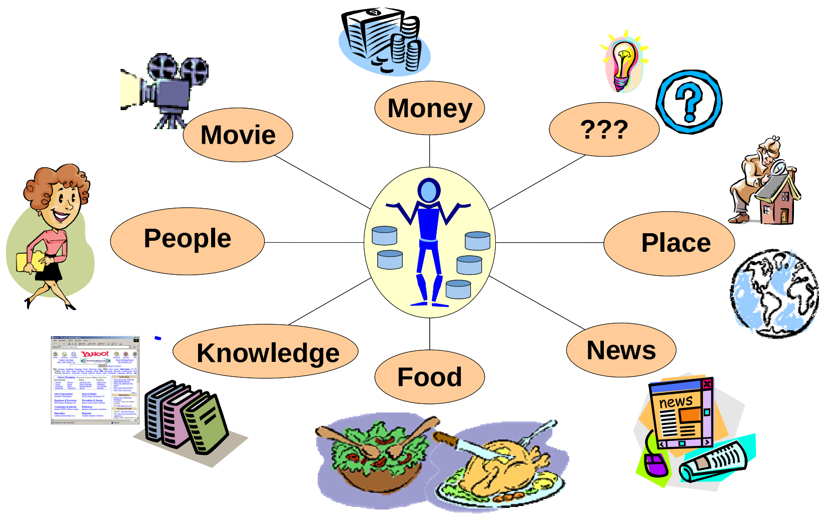


Figure 2-2: Individual communication space & underlying physical resources

The difference between *context* and *active context* is characterized by the entities, which are considered in relations and causalities. Context only refers to objects as an abstract model of what kind of objects have to be taken into account in a certain context, whereas an active context refers to physical resources that have been identified during the activation process. Active contexts are of dynamic nature reflecting the current environment an individual resides in.

The activation and deactivation of contexts is one task of I-centric services. To activate a context the I-centric service performs the activities described above. In addition, the I-centric Communications system has to manage concurrent access to objects and conflicts caused by contrary wishes, expressed by individual(s).

Active Context: A context is active when it is adapted to a certain environment at a certain moment in time. It defines the relationships and causalities of an individual to a particular number of physical resources at certain a moment in time, in a certain environment.

Deactivating a context means:

- The identification of objects that have been activated/controlled in the context
- The identification of affects that have been caused on other objects by the context
- The identification of activities (e.g. deactivation, reconfiguration) that have to be performed on all these objects and physical resources underneath for deactivating the context
- To perform these activities on the objects and physical resources

Acting in a context means to use only services that are provided by objects, which are part of that very context. Starting to interact with objects that are not part of an active context will cause the activation of another context. That means, on one hand individuals are allowed to act in several contexts in parallel, and on the other hand, the I-centric communication system must handle conflicts that might occur due to contrary causalities defined in the different contexts.

To handle each individual communication space and associated contexts, a general model of domain information and relationships to objects and physical resources is needed. This model must be flexible to be enhanced due to the introduction of new locations, devices, etc. Furthermore, the model has to provide mechanisms to map objects to physical devices to support the activation and deactivation of contexts.

2.2.2 Preferences and Ambient Information

Individuals have different *preferences* in different situations. Sitting alone in a silent room might indicate that an individual is willing to receive incoming phone calls. However, the same individual can perceive this as a disturbance when being involved in a conversation with others. With preferences, individuals express their choices of services characteristics in certain contexts. Therefore, preferences provide a powerful mechanism to influence the behavior of I-centric services by giving them explicit instructions. Considering the example above again, the individual might have the preference not to be disturbed by incoming phone calls in such situations.

Preferences: Preferences are conditional *choices of service characteristics of an object* depending on context and ambient information. Preferences are applied to objects during the activation of a context.

I-centric services evaluate preferences to adapt their behavior to what is ‘really wanted’ by an individual in a certain environment at certain moment in time. Therefore, preferences have to be either gathered from individuals interactively or automated by monitoring, and they have to be expressed in a machine computable form.

Gathering preferences interactively causes a lot of interaction between the I-centric system and an individual, who could feel bothered after a short period. A more desirable approach is the collection of preferences based on monitoring the behavior of an individual over time. Inference mechanisms can be applied then to the collected data to predict what preferences a certain individual might have. This process has to run continuously to keep the preferences up-to-date or to precise them step-by-step.

The description of preferences, which can be processed automatically, is another challenging task. Preferences can capture many aspects like mood, interests, live stage etc. that are even hard to describe in words. Furthermore, the kind of preferences that are relevant to different individuals may differ completely. A model for describing preferences must be as generic as possible to avoid restrictions that might prevent the expression of a certain preference. On the other hand, the model has to provide some structuring or categories to allow the assignment of preferences to a certain I-centric service.

Preferences should be as precise as possible. One approach to teach this is to relate preferences to contexts and to information describing environmental conditions. Continuing the example given in the beginning of this section, only the existence of environmental conditions allows the specification of ‘a silent room’. Beside the noise level, many other environmental conditions might be sensed by I-centric communication systems. To emphasis the variety of such kind of information, the term *ambient information* has been coined by the ISTAG group in its ‘Scenarios for Ambient Intelligence for 2010’ [ISTAG].

In general, ambient information is information that can be collected, gathered, or sensed from the environment. Ambient information comprises temporal and spatial characteristics like any user input, temperature, noise level, light intensity, and presence of other people just to give a few examples. Ambient information is sensed by sensing facilities, like motion detectors or microphones, and transmitted through sensor networks. Ambient information may also include geographical information (e.g. location), environmental information (e.g. temperature), and life conditions (e.g. blood pressure).

<u>Ambient Information:</u> Ambient information is information that can be collected, gathered, or sensed from the physical environment using the objects of the individual communication space of a certain individual.

Furthermore, ambient information is the basis for ambient-awareness, which is introduced in section 3.4. A semantic model is needed to describe preferences and ambient information. Such kind of model incorporates knowledge representation to qualify available information and ontology languages to relate syntax and semantic to each other (e.g. Semantic Web [WWWSEMW], Agent Ontology Service [WWWAOS]). The focus for I-centric Communications here is to define a harmonized semantic model that includes human aspects as well as the process to gather, store, evaluate, and exchange preferences as well as ambient information.

2.2.3 I-centric Service

I-centric Services define, manage, and (de)activate contexts in an individual communication space taking the preferences of individuals and ambient information into account. They support an individual (I-centric), adaptive, personalized, and ambient aware way to interact with objects in individual communication spaces. Based on the evaluation of ‘profiles’ that describe preferences, service capabilities, and sensed information about its actual environment, the individual can be provided with personalized services for his actual demands.

<u>I-centric Service:</u> I-centric Services define, manage, and (de)activate contexts in an individual communication space taking ambient information and the preferences of an individual into account. They support an adaptive, personalized, and ambient aware way to interact with objects in individual communication spaces.

Self-learning capabilities are used to profile the behavior of individuals. Numerous services or several features of different services are combined on-demand. Appropriate terminals and adaptation strategies are evaluated.

I-centric services need ambient information in order to adapt to the environment. Temporal and spatial characteristics are only two examples of information, which may affect the service behavior. Note, that a certain environment can restrict the functionality requested in a certain context. Interacting in a ‘TV context’ while driving a car may reduce the available functionality to ‘*record the movie for later viewing*’ or to listen just to the audio part.

I-centric services activate contexts by choosing the equipment to be controlled (presentation terminals, handhelds, microelectronic controlled devices), their quality of service (volume, brightness, etc.) to be finally connected via heterogeneous networks (BAN, PAN, LAN, WAN) to create an I-virtual private network (see section 2.2.1).

The process of choosing and controlling the equipment of the physical environment is supervised by the service logic of I-centric services. The service logic controls the activation of contexts by combining multiple objects dynamically. It parameterizes objects by defining what, when, and how one or more objects behave in a given condition. The service logic decides based on profiles and on the status of the objects how those objects should behave in a certain situation. This enables sensitive services that adapt to the environment dynamically. Therefore, mechanisms have to be developed, which enable to gather and to evaluate profiles. The profiles and the status will be evaluated by a domain-specific service logic that also controls the object(s).

Nowadays, service logic is in most cases ‘hard-coded’. Once implemented, it cannot be changed afterwards. The basic idea of I-centric Communications, to provide individuals with their *own* services that might change over time, requires a more flexible approach. A generic model for describing service logic independent from any execution environment is needed. The description of service logic has to be an input parameter, like ambient information, during the execution of an I-centric service. That will allow altering the service behavior according to changing situations. Moreover, creating new or altering existing service logic can become an interactive or automated process. Self-learning systems with automatic reasoning engines or interactive tools can be applied then.

The process of creating or modifying I-centric services has to be accompanied by ontology definitions that describe what services an object is providing. Interactive applications are envisaged that allow individuals to assemble their service by simple ‘drag and drop’ mechanism. Like a LEGO™ toolbox, the individual should be able to create and to deploy its I-centric services.

2.3 Reference Model for I-centric Communications

Following the vision explained in section 2.1, this section introduces the reference model for I-centric Communications. The work on the reference model has been accompanied by three international activities. They are described in the end of this section.

Figure 2-3 shows the reference model for I-centric Communications. It follows a top-down approach starting with the introduction of *individual communication spaces*, related *contexts*, and *objects*. In general, the topmost layer recalls the I-centric vision that human beings interact with objects of their communication in a certain context. Furthermore, it is common understanding that I-centric services have at least to support three different features, namely *ambient awareness*, *Personalization*, and *adaptability*, which will be explained in sections 2.3.2, 2.3.3, and 2.3.4 in detail. To emphasis that these features are needed for I-centric Communications they have been assigned to the individual communication space in the reference model.

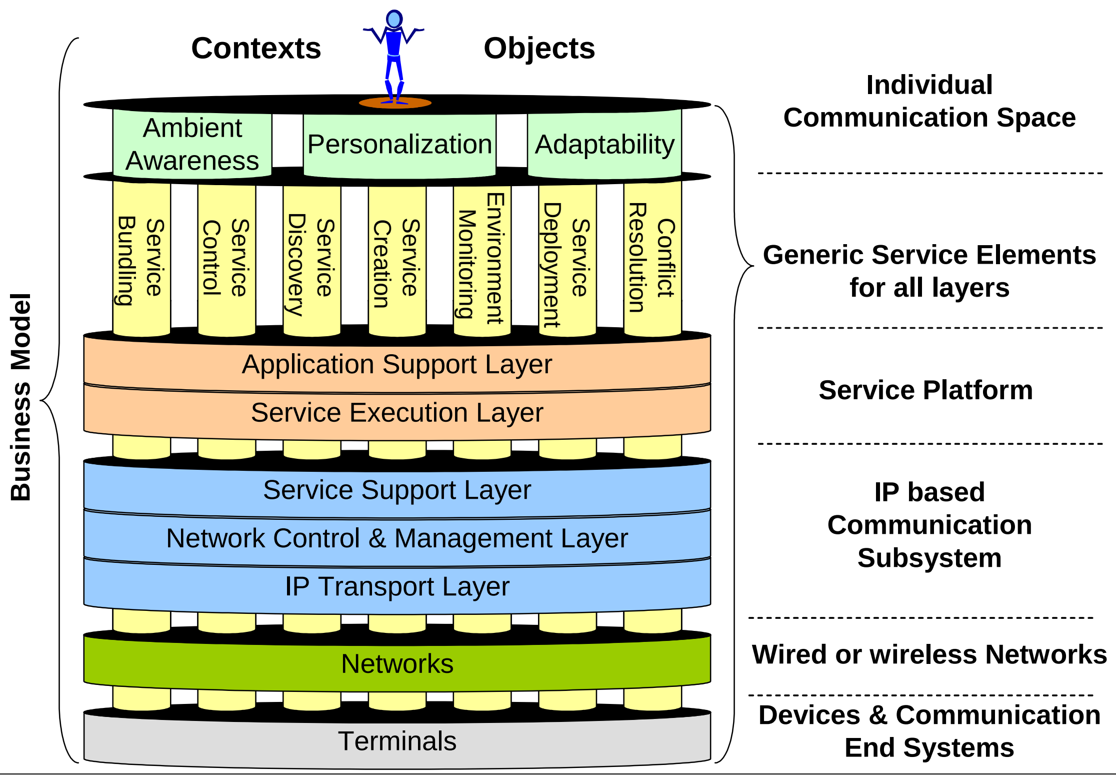


Figure 2-3: Reference Model for I-centric Communications

The *service platform* for I-centric Communications is responsible for shaping the communication system, based on *individual communication spaces*, *contexts*, *preferences*, and *ambient information*. Preferences will be provided by the personalization feature, whereas ambient information has to be provided by ambient-awareness feature. More details are given in section 2.3.5, which introduces the service platform for I-centric Communications.

The *IP based communication subsystem* is responsible for providing the linkage between different objects in the communication spaces. These links have to be maintained and managed even when they are subject to change because of roaming between different network topologies or access networks. There might be non-IP based communication networks underneath the IP-based communication subsystem. The have to be wrapped by bridging facilities (e.g. by SmartIP devices[4]) to include them in I-centric communication systems. IP communication is seen as the common denominator to harmonize heterogeneous network infrastructures. The IP based communication subsystem consists of three layers:

- Service Support Layer, which provides well defined APIs for the service platform to access the IP based communication subsystem
- Network Control & Management Layer, which combines the traditional concepts of network management with required real-time aspects needed for system wide control functions
- IP Transport Layer, which basically represents OSI layer four

The *Wired or wireless Networks layer* implements all aspects of the physical connection(s) between different objects. Due to the hierarchical structure of the reference model, a connection

[4] SmartIP devices introduce new kinds of computing facilities into the IP world. Nearly each kind of device can be equipped with IP technology enabling the global accessibility of these devices. Even large groups of devices using proprietary networking technologies can be wrapped by a single SmartIP device providing address translation to reach each of the non-IP devices.

in the IP based communication subsystem might use multiple connections in underlying network.

Devices and Communication End Systems provide the physical infrastructure that hosts all other layers. It can be instantiated as a switch responsible for connecting different networks or even as a multi-modal terminal able to interact with a certain individual.

The main features of I-centric Communications (ambient-awareness, Personalization, and adaptability) affect all layers. Therefore, supporting functions have to be provided as a vertical solution. The reference model introduces the concept of *Generic Service Elements* that implements common functionalities on all layers. Generic Service Element can be seen as a toolbox from which complex services can be assembled and executed dynamically. The vertical approach allows I-centricity on all layers, e.g. to establish I-centric private virtual networks (section 2.2.3). More details of Generic Service Elements are given in section 2.3.6.

Accompanying to all these technical issues, the *Business Model* for I-centric communication identifies the relationships and information flows between all active roles within an I-centric system. The business model will help to identify reference points between all involved entities and will assist the assessment of the applicability of I-centric services for different business domains.

The IP based communication subsystem, the wired/wireless networks, and all terminal aspects are out of the scope of this thesis. The work is based on the assumption that the APIs provided by the Service Support Layer will cover all necessary functionalities to perform the tasks of I-centric Communications. Therefore, the service platform defines a set of requirements to these APIs (for more details see section 2.3.5).

In the following, the building blocks of the reference model for I-centric Communications will be discussed in more detail.

2.3.1 Business Model

The vision of I-centric Communications requires new business models. The borders between traditional roles and administrative domains: network provider, content provider, service provider, and retailer are blurring. An individual may become service provider (ad-hoc and peer-to-peer), or content provider, or retailer. Additionally, roles may change dependant on context, which implies a very flexible business model.

The business model for I-centric communication has to cover:

- Roles, relationships, and reference points
- Business topologies
- Service lifecycle (creation, deployment, management, and billing)
- Benefits for parties involved in the market value network

Business Model: A business model is a description of how an entity or a set of entities intend to create value with a product or service. It defines the product or service, the roles and relations of the entity, its customers, partners and suppliers, and the physical, virtual and financial flows between them. [WWWWG2BM]

The first objective of a business model for I-centric communication is a model for describing relationships between involved parties in a global business community. Based on these relationships, roles and reference points will be defined. This allows the participation of each business partner on a global business on one side and provides the freedom in development and integration on the other side. Reference points provide standardized points of contact and information exchange between business partners. Such a business model for I-centric Communications is a prerequisite for the definition of a service architecture that supports the required functionalities of the whole business life cycle.

2.3.2 Personalization

Personalization is considered as being the key factor for I-centric Communications. Information and services must become increasingly tailored to individual preferences to make the usage of services easier and the perception of the individual communication space richer.

An extended personalization concept is needed that enables value networks (e.g. value chains) of content providers, network providers, and service providers to offer personalized services to mobile individuals in a way that suits their needs, at a specific place and time.

Personalization aspects like preferences, role, and task, have to be integrated and the relation between the aspects must be studied. Investigations into benefits according to the user perception are necessary and needs to be considered in the design of personalized services and personalization supporting frameworks.

For I-centric Communications, this means that objects available in an individual communications space have to adapt to the preferences of individuals. Personalization models each individual in the I-centric service platform by managing its preferences and providing these preferences to I-centric services.

<u>Personalization:</u> Personalization provides the information for modeling preferences for an individual communication space in the I-centric system.

To reach this goal, personalization federates profile information (containing preferences). Personalization incorporates dynamic behavior to enrich stored and federated information to enable pro-active I-centric services. This leads to an overall profiling infrastructure managing the individual preferences.

The main research issues behind personalization are:

- How to gain personal preferences from individuals (interactively or automated)
- How to store these preferences in profiles (profile format & categories)
- Standards to exchange profiles
- Secure privacy sensitive parts of profiles
- How to describe active contexts using individual preferences

2.3.3 Ambient-Awareness

In I-centric systems, services will be tailored to contexts of the individual communication spaces. The services will automatically adapt themselves to changes in the environment. The services have to deal with a dynamic environment of nomadic individuals. The adaptation to the situation (in a certain context) is hidden from the individual. It is provided with optimal experiences and benefit. Vice versa, the environment can be influenced by the presence and activities of the individual and adapt itself accordingly.

Therefore, ambient-awareness deals with gathering ambient information from network, application, individual, terminal, and contexts. The federation of ambient information from various sources, according to individual's mobility and roaming is an integral part of ambient-awareness. Intelligent inference systems for missing information are needed in order to incorporate as much information as possible to provide an automatic, ambient aware environment to the individual.

Ambient-awareness: Ambient-awareness is the functionality provided by an I-centric system to sense and exchange information about the current environment, an individual is in at a certain moment in time.

Sensors networks will play a major role in providing ambient information. Sensor technologies will be embedded in the mobile equipment, communication networks, living and working environments to sense who the user is, where he is, what he is doing, what the environmental conditions are, to provide this ambient information to I-centric services.

Advances in sensor technology are conditions to reach further adaptation of services to the environment of individuals. Many devices in our current world already adapt in some form to their operating environment. One example is the television set that adjusts its image contrast to the ambient lighting level. 'Intelligent I/O behavior' can be used to adapt the output characteristics depending on the context, but also the input characteristics should change in many situations (do not bother the individual with long lasting interactions, but combine the context information with the basic intention that was beforehand communicated by a few interactions). This intelligent I/O behavior demands the development of multi-mode user interfaces and corresponding support functions in the various consumer devices as well as in the service adaptation network.

2.3.4 Adaptability

Section 2.3.2 (Personalization) and section 2.3.3 (Ambient-awareness) have introduced two main functionalities of I-centric services. Adaptability is the third one and is mainly based on information provided by personalization and ambient-awareness. It provides the functionality to adapt I-centric services to personal preferences and environmental conditions. Therefore, adaptability can be seen as a function that activates a context based on whatever information is provided by ambient-awareness and personalization.

In general, I-centric adaptability translates the wishes of individuals, which are usually inaccurate, incomplete and sometimes even contradictory, into a set of rules precise enough for processing to be automated with sufficient reliability. It has implications in the structure of the services to allow adaptability and is the engine, which activates a context at a certain moment in time in a certain environment. [WWWWG2AD]

Adaptability: Adaptability is the functionality provided by an I-centric service to activate context taking ambient information and individual preferences into account. An active context might be adapted continuously to reflect changes of individual preferences or environmental conditions over time.

Typical situations when adaptation takes place include a substantial change in characteristics of connectivity, entering into a new service domain, or changing terminal device in Service Session.

By technical means, adaptability requires the adaptation of media, content, and service behavior. During the last years, a variety of concepts for adaptation has been developed [ICS1, ICS15, ICS16, PFEIFER1]:

- Communication streams can be altered during transmission (e.g. bit rate adaptation),
- Media types can be changed (e.g. text-to-speech conversion),
- Type of presentation can be adapted (e.g. downscaling an image to fit a PDA screen),
- Altering the content of a message (e.g. adding or stripping off information), or
- Modifying the service behavior (e.g. by customer service control functions).

Adaptability cannot be only reactive. When the battery of a mobile device dies or the connectivity breaks, many actions become impossible. However, something could have been done beforehand. Therefore, adaptation must also be proactive, which in turn requires predictability of the near future. An important question in predictions is to distinguish between the situations in which the human behavior seems to be predictable and those being unpredictable.

2.3.5 Service Platform for I-centric Communications

A Service Platform for I-centric Communications is responsible for shaping the communication system, based on individual communication spaces, contexts, preferences, and ambient information. Finally, it (de)activates objects (advised by I-centric service), identifies causalities between them based on sensed environmental data, controls the services offered by these objects, and converts data structures and operations for interworking between services. The equipment is configured dynamically, its state is profiled, distributed objects are controlled, service creation and deployment are supervised, and the interworking among domains is enabled by the platform.

Service Platform: A service platform is an infrastructure that supports the development and operation of I-centric services by providing a set of service features:
- Execution environment for services and objects
- Supports (re-)deployment (hot-plugging) of services and objects
- Supports the (re-)binding (configuration) of services and objects
- Supports the interworking of services and objects

The service platform consists of two layers:

- The Application Support Layer provides well-defined APIs to applications, services, and objects. It offers generic service elements that can be used by developers of these entities to ease and fasten the process of design, implementation, deployment, and management.
- Service Execution Layer provides the actual runtime environment of applications, services, and objects. It supports their secure, QoS aware and managed execution.

Moreover, the service platform provides functional blocks that directly support the I-centric approach. These functional blocks manage ambient information, preferences, and adaptability to be offered to I-centric services. To fulfill the functionalities requested by I-centric Communications, I-centric service platforms have requirements on the underlying communication subsystem. This is caused mainly by the empowerment of any individual to act as a service provider or network provider in a paradigm shift from a provider centric paradigm to a decentralized I-centric paradigm. From an I-centric perspective, this is done by sharing objects between different individuals or by allowing another individual to use objects out of ‘my’ individual communication space.

On the other hand, the requirements are based on information that has to be provided by lower layers to the service platform. Traditional platform approaches (e.g. object-oriented middleware platforms) try to hide as much as possible technical parameter between the different layers. I-centric services have to be provided with ambient information. Therefore, the traditional concept of transparencies [ODP, X.901-904] vanishes. Ambient information has to be provided trough all layers. Only the I-centric service itself can decide what information is useful or not. Intelligent filtering mechanisms and translation rules will help the I-centric services to ‘understand’ the ambient information.

By technical means the requirements of the service platform to the underlying network are:

- Peer-to-peer, ad-hoc communication (single or multi-hop, ad-hoc relays)
- Bearer technology aware stacks and applications
- QoS in various areas (guaranteed streaming, data-rate, real time traffic)

- Global roaming by integrating access networks supporting various wireless and wireline bearers, various radio interfaces, and mobile-IP
- Ubiquitous Addressing (Address mapping, Routing over Address-less networks)
- Multicast services (incl. advanced addressing methods like geo-cast)
- Personal Identifiers to overcome address schemes of traditional communication networks
- Sensor networks for gathering ambient information
- Privacy, assurance of the confidentiality of data
- Integrity, assurance that the content of a transmission cannot be altered during transmission
- Confidentiality, assurance of the confidentiality of information
- Non-repudiation, assurance that the sender (receiver) cannot deny having sent (received) information
- Authentication, assurance of the identity of the sender and the receiver of information
- Access control/authorization, Assurance that only authorized people can access information
- Accounting/billing. Transparent billing for service usage and integrated network access

However, the paradigm shift addressed here does not only concern the individual as a provider of network related services. A service platform allowing global mobility and transparent access to any kind of service over a common IP platform is the basis for allowing everyone to provide a wide range of services [WWWWG2BM].

2.3.6 Generic Service Elements

I-centric Communications systems will have to cope with issues like numerous service providers, always-connected individuals, automatic service adaptation, and ambient-awareness. Aspects like dynamic service discovery and service provisioning in (for individuals and services) unknown environments and personalized services usage requires new mechanisms to support I-centric Communications systems.

To simplify the definition and realization of I-centric services and applications, a set of reusable software components will support functionalities common for different services and applications. These components are called Generic Service Elements to emphasis their general applicability for all kind of services.

Generic Service Elements: A GSE is a functional software component that can be used by other GSE's, services, or applications and it is hosted by the I-centric service platform. GSEs provide functionalities common to different services and applications to ease and shorten their development process.

Because I-centric services should work under changing environmental conditions, serving changing individual preferences, the most promising candidates for common functions are:

- Service Discovery (a mechanism to discover service features dynamically that are provided within a certain environment or by a certain physical resource)
- Service Management (how to manage context (dynamic relationships and causalities between individuals and their environment))
- Service Deployment (how to deploy services in distributed environments)
- Service Composition (dynamic interworking of services will help to create and operate contexts)

- Service Logic (the evaluation of the preferences and ambient information leading to a decision what has to be done by an I-centric services)
- Service Control (the process to control all the resources needed for a specific purpose)
- Environment Monitoring (how to gain ambient information)
- Reservation (to manage exclusive usage of objects)

By technical means, objects and generic service elements are quite similar. They have to provide well-defined interfaces to be used for service developments. Generic service elements can also be seen as enabler for objects by providing functionalities that are common for all objects.

Consequently, well-defined collections of interface specifications designed for certain business domains are needed. The idea is to equip same kinds of objects with standardized interfaces for functional (usage) and non-functional (management) interfaces. Especially, from the area of telecommunication, Open Service APIs like OSA/Parlay [WWWOSA] build the basis for such interfaces. As the vision for future mobile systems does not cover communication scenarios only, new Open Service APIs have to be developed. On the one hand, the service infrastructure should be opened by means of Open APIs to enable easy of use service creation. Such APIs must provide framework functionality, like hot plugging of services, dynamic (re-)binding (e.g. in ad-hoc environments), service mobility, AAA-services, support for automated SLA negotiation, contracting, and so on. It is to remark, that this generative approach may require well-defined internal API's and callbacks, too.

2.4 State of the Art

In the following, four activities are briefly introduced, which have influenced the work on the reference model for I-centric Communications. Namely, they are the Unified Messaging Research project, the IST projects WSI and WWRI, as well as the activities ongoing in WWRF's Working Group 2.

2.4.1 Unified Messaging Research Project

The concept of Unified Messaging has emerged from the research of Personal Communication Support [ECKARDT1]. Reaching industrial relevance it addresses the task of overcoming the multiple-mailbox approach of today's communication scenarios with separated facilities for e-mail, voice storage, fax reception, etc. This coincides with the vision of I-centric Communications, to deliver information any time, any place, in any form, as it is also described in the concept of the UMTS Virtual Home Environment [ETSITS22.01]

A unified messaging system has been implemented within the Unified Messaging research project[5] [ARBMEER1]. The project has addressed service access and service delivery on both, fixed and mobile terminals. To support this variety of end systems, a special capability of the developed messaging system is the selection of the most appropriate terminal or application for an incoming message. The most appropriate terminal will be determined by means of its availability, status, and capability to handle a certain communication service. If no appropriate terminal can be found, a pool of converters adapts the communication media to the available terminal on demand.

The provision of Customer Service Control Capabilities, enabling users to define rules for the handling of incoming calls and messages according to their individual preferences, is another important feature of the system. The messaging system acts as an electronic secretary. Once

[5] The project was a research collaboration between the Fraunhofer Institute FOKUS and the Department for Open Communication Systems at the Technical University of Berlin.

advised by the user, it is responsible for controlling the user's communication environment, i.e. telephone calls or e-mails. Each service is processed based on user-predefined rules.

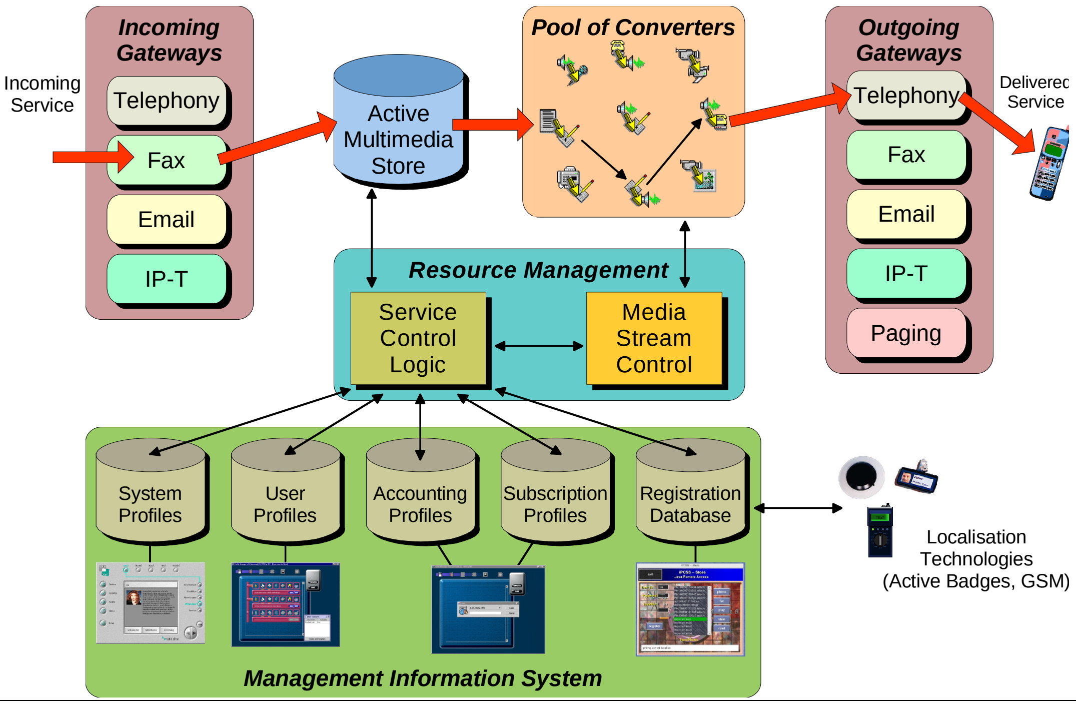


Figure 2-4: OKS' Unified Messaging Systems Functional Components

The messaging system incorporates the following features:

- Call Screening, Call Scheduling, Call Forwarding
- Customer Service Control
- Adaptation of Services and Automated Content Adaptation
- Multimedia Message Storage and Remote Message Access
- Synchronous and Asynchronous Service Delivery
- User Registration at Terminals and Locations (automated, manually, or scheduled)

To be able to adapt easily to new bearer networks, services, and applications, the messaging system separates service access, service logic, and service control. Specific parts of the system should be responsible for the access to particular bearer networks, while generic parts take care of the service logic and the service processing (the establishment of bearer connections including necessary service conversions).

A *Customer Service Control Component* allows users to configure their individual preferences using a rule based *Service Control Matrix*. In rules a user can define when, where, for whom, and using which medium or terminal he wants to receive a message. A setting likes 'In the next two weeks, my mobile phone with the number 123456 should only ring if my girlfriend wants to reach me', could be described in two rules:

- Between day X and Day Y, if user Z calls me → forward to my telephone 123456.
- Between day X and Day Y, if calling user is not Z → forward to my voice-mail-box.

A user can have several rules. The rules of a user will be evaluated for each incoming communication request. The results of an evaluation directly affect the *Service Control Component*, which handles the communication request as the user has configured it.

The realized Unified Messaging System implemented some functions that have been taken into account during the specification of the architectural framework for I-centric Communications (e.g. Customer Service Control and selection of appropriate terminals, for certain communication requests).

2.4.2 IST Project Wireless Strategic Initiative

The reference model developed by the Wireless Strategic Initiative (WSI) [WWWWSI, WSIREF] describes the Wireless World as a set of concentric spheres inhabited by networked Communication Elements (CEs). CEs are the generic representation of devices and nodes in the Wireless World. The functions incorporated in a CE are provided by different building blocks.

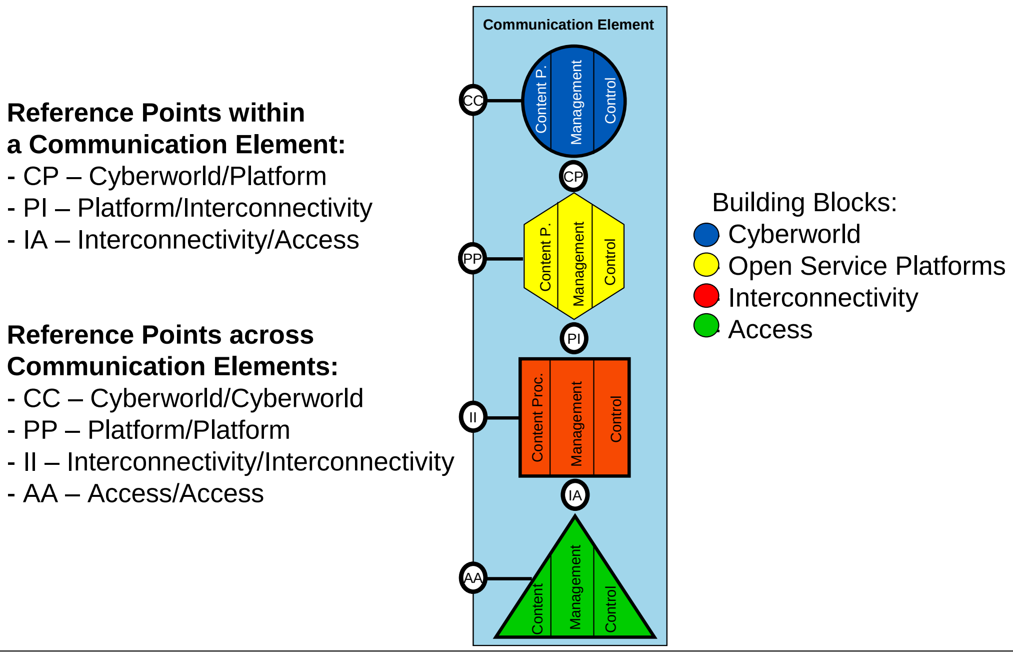

Communication Element
Content P.
Management
Control
CC
CP
PP
PI
Content Proc.
II
IA
Content
AA
Reference Points within a Communication Element:
- CP – Cyberworld/Platform
- PI – Platform/Interconnectivity
- IA – Interconnectivity/Access
Reference Points across Communication Elements:
- CC – Cyberworld/Cyberworld
- PP – Platform/Platform
- II – Interconnectivity/Interconnectivity
- AA – Access/Access
Building Blocks:
Cyberworld
Open Service Platforms
Interconnectivity
Access

Figure 2-5: WSI Reference Model

The identified building blocks are namely:

- The **Cyberworld** block, hosts all application-specific functions and manages their characteristics to ensure the underlying infrastructure is being used efficiently to satisfy the user.
- The **Open Service Platform** block, is responsible for providing a service infrastructure to facilitate the creation of services according to user and operator needs.
- The **Interconnectivity** block is the Networking part of the Wireless World reference model. The functions located there take care of linking together resources.
- The **Access** block implements all aspects of physical connections between different entities.

Reference points have been identified between the building blocks of a Communication Element and among different Communication Elements. The reference model for I-centric Communications has been developed in parallel to the WSI project. Therefore, the two reference models are very similar. The I-centric model is focusing more on the user perspective, whereas the WSI model was more oriented on the networking and radio aspects.

2.4.3 IST Project Wireless World Research Initiative

The strategic objective of WWRI project [WWWWRI] was to provide a launch pad to the wireless community (industry and academics) for the development of a balanced cooperative research program addressing the future Wireless World.

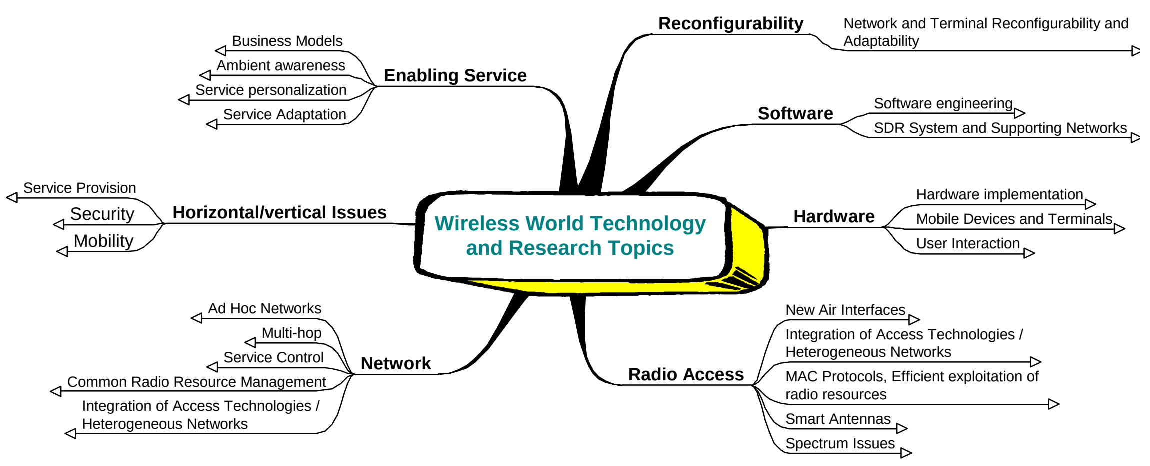

Wireless World Technology and Research Topics
Reconfigurability
Network and Terminal Reconfigurability and Adaptability
Software
Software engineering
SDR System and Supporting Networks
Hardware
Hardware implementation
Mobile Devices and Terminals
User Interaction
Radio Access
New Air Interfaces
Integration of Access Technologies / Heterogeneous Networks
MAC Protocols, Efficient exploitation of radio resources
Smart Antennas
Spectrum Issues
Network
Ad Hoc Networks
Multi-hop
Service Control
Common Radio Resource Management
Integration of Access Technologies / Heterogeneous Networks
Horizontal/vertical Issues
Service Provision
Security
Mobility
Enabling Service
Business Models
Ambient awareness
Service personalization
Service Adaptation

Figure 2-6: Overview on technology and research areas structuring

The concrete tasks have been to produce a report on the wireless industry sector and its likely developments in order to be able to establish the strategic research directions for the wireless sector in the timeframe until 2010, and to establish and propose technical areas of research (Figure 2-6) as well as an evolution roadmap for systems 3G beyond.

The results have mainly influenced the orientation of I-centric communication towards ambient-awareness, personalization, and adaptability. These issues have been identified as mandatory for future services based on a broad consensus between mobile operators, manufacturer, and academics.

2.4.4 Wireless World Research Forum

The Wireless World Research Forum (WWRF) [WWWWRF, MOHR] has the objective of formulating visions on long-term strategic research directions in the wireless field, involving industry and academia. The aim is to generate, identify, and promote research areas and technical trends for mobile and wireless system technologies. In the 'Book of Visions 2001' [WWWBOV], the forum has identified needed research items towards wireless systems 3G and beyond.

Working Group 1 The User in the Driving Seat	**Working Group 2 Service Infrastructure of the Wireless World**	**Working Group 3 New Communication Environment/ Heterogeneous Networks**	**Working Group 4 New Technologies**
Main research tasks: ❑ Understanding the user ❑ New generic application elements ❑ New interaction techniques	**Main research tasks:** ❑ Business models ❑ Personalisation ❑ Ambient awareness ❑ Adaptability ❑ Generic service elements ❑ Enabling technologies	**Main research tasks:** ❑ Wireless World networking architecture ❑ IP mobility architecture ❑ QoS architecture ❑ IP Multicast architecture ❑ Transport protocols ❑ SDR	**Main research tasks:** ❑ Spectrum issues ❑ Antenna arrays and related techniques ❑ New air interfaces ❑ Ad hoc networks

Figure 2-7: WWRF – Working Group Structure

The vision of I-centric communication has been adopted by the WWRF. It builds the basis for the definition of the service architecture for the Wireless World. Especially WWRF working group 2 is investigating service architectures that support I-centric applications.

2.5 I-centric User Interaction

As described in the topmost layer of the reference model for I-centric Communications (section 2.3), human beings interact with objects of their communication space in a certain context. As shown in Figure 2-8, a *user interface* facilitates this interaction. From the perspective of I-centric Communications, there is a ubiquitous user interface surrounding the individual and providing a suitable information exchange with the various objects of the individual's communication space. As the concerned objects are defined by a given context, the interaction with these objects should consider this context, likewise. The objective of I-centric User Interaction is to realize this particular user interface by following the vision of I-centric Communications.

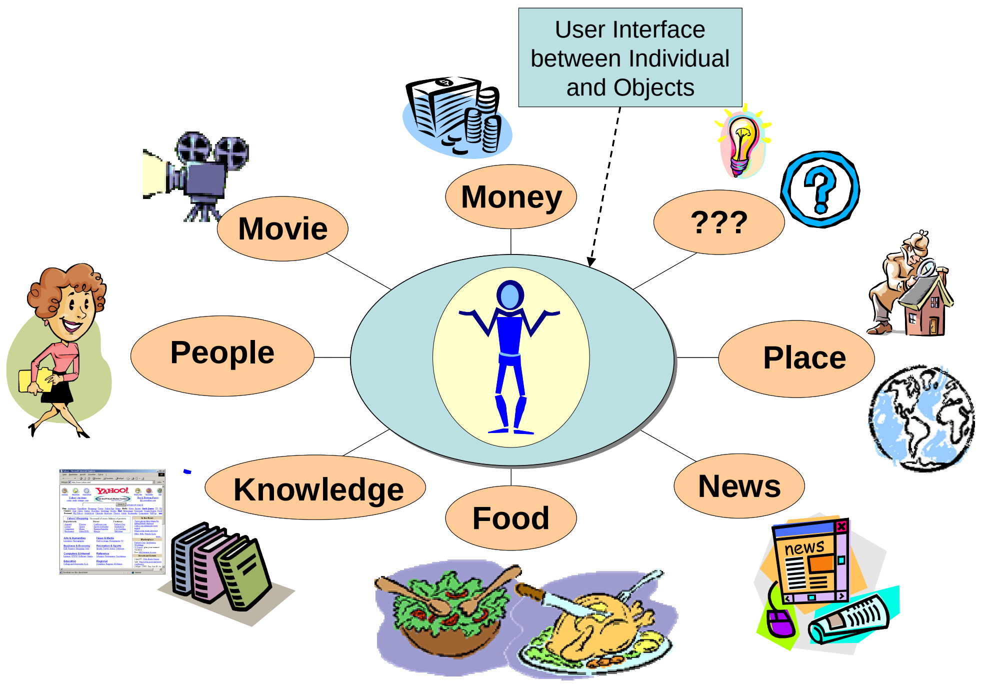


Figure 2-8: Interaction with Objects in the individual Communication Space

The interaction with the objects is not necessarily a direct interaction with the corresponding underlying physical resources. Instead, the interaction with abstract objects, such as 'food' or 'money', can be realized through according services that manages these objects. However, from an individual's point of view, the individual interacts with the objects, irrespective how the interaction is realized in detail.

Instead of starting to analyze the particular input and output capabilities of all possible objects in the individual communication space, I-centric User Interaction starts with analyzing the interaction behavior of human beings. This depicts a user-centered approach as required by the vision of I-centric communication. The goal is not to develop completely new kinds of technologies to implement user interfaces, but to develop a generic approach, which abstracts from technical details and which is therefore not limited to specific user interface technologies. This generic approach should be capable to support any kind of user interface technology.

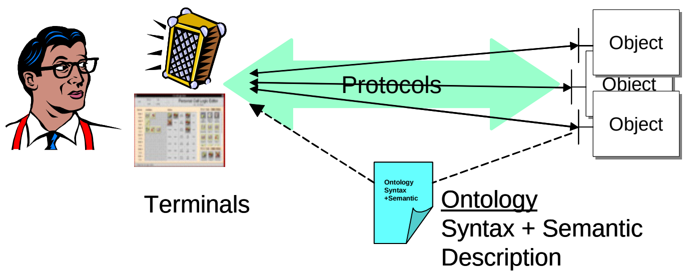


Figure 2-9: Interaction with Objects

I-centric User Interaction has to define the necessary concepts to enable the interaction between the individual and the abstract objects. As shown in Figure 2-9, suitable protocols to describe the interaction and to realize the user interface on the user's terminals are required.[6] The realization of the user interaction should consider the different capabilities of terminal and network, the user's preferences as well as the user's situation, which is described by ambient information. The concepts should comply with the general vision of I-centric communication and should reuse existing approaches and standards. This thesis introduces a suitable approach to realize I-centric User Interaction (see chapter 3). For the development of this approach, some assumptions and requirements have to be considered, which are introduced briefly in the following.

The interaction of human beings can be *implicit* or *explicit*. Both kinds of interaction are integrative part of the vision of I-centric Communications. An I-centric system should provide both interaction schemas seamlessly realizing an environment in which technologies are almost imperceptible by humans.

- *Implicit interaction* means that the system follows the activities of the individual by analyzing the available ambient information. With them, the services can determine the user's current context and can react accordingly. The user does not have to be aware of this interaction. The individual is rather an 'actor' than a 'user'. That is why this kind of interaction is described as *implicit* user interaction.
- In the *explicit interaction*, the user interacts with the services in a conversation. The user can instruct services by giving suitable commands, e.g. to activate a certain context. This conversation can be a dialog between user and service, where the user provides input and the service responds to it accordingly. For this interaction, the user needs one or more terminals, which realize the user interface to the services and which provide the necessary input and output capabilities.

The concept of I-centric User Interaction concentrates on the *explicit interaction* between individuals and objects of the individual's communication space. According to the general approach of I-centric Communications, this concept defines the criteria, models, and an architectural framework for I-centric User Interaction.

As described in the vision for I-centric Communications (section 2.1), individuals are interested in the semantic and not necessarily in the presentation of services. The interaction with objects in an individual's communication space has to support the quality of the human senses, and since quality of senses is individual, the objects have to adapt their presentation to each individual automatically.

The I-centric User Interaction approach continues this development by improving the user interaction by means of intelligent adaptation capabilities. The user shall interact with the objects in the individual communication space in a comfortable and easy-to-use way, independent from location, time, and available terminal devices. In contrast to classic approaches, the user interface shall not restrict the user in the interaction, rather the interaction shall adapt to the user.

[6] These requirements and according approaches are presented and discussed in more detail in [ICS6].

Adaptation of the user interaction means that the most suitable terminal and communication service is selected and that the interaction is adapted to the respective capabilities and according to the current context as well as to the individual preferences. In this way, individuals are not restricted to a particular user interface technology in the interaction with the objects of their respective communication space.

As identified in the beginning of section 2.1, the services, provided by objects in the individual communication space, shall adapt their presentation to the contexts and the preferences of the individual. The context also describes here the situation in which the interaction takes place (described by ambient information). The I-centric reference model identifies the ambient awareness functionality, which describes the capability to sense and exchange information about the current environment, the individual is in. Therefore, this functionality provides sensible information, which should be considered for the adaptation. In addition, the consideration of the context describing all the relationships an individual has to certain objects provides a useful base for an adaptive interaction.

This implies that I-centric User Interaction has to support personalization and ambient awareness by intelligent adaptation capabilities. This adaptation feature will make the user interaction more comfortable and powerful, which will in turn lead to a higher degree of acceptance of I-centric services by users. As the vision of I-centric Communications orientates on the human and the actual needs, the I-centric User Interaction aspect tries to improve the user interaction by following and imitating the capabilities of human-human interaction.

This means, I-centric User Interaction has to provide a suitable support to realize the interaction between individuals and the objects in their individual communication space. The support has to comprise the reasonable adaptation to:

- The user’s preferences to enable a personalized appearance of the user interaction
- The current environment of the user to enable an ambient aware appearance of the user interaction
- The particular terminal and network used in the interaction to enable Device Independence, i.e. technology independence

These areas of concern are interconnected and have to be considered coherently in order to realize a sustainable I-centric User Interaction system. Accordingly, the approach to realize I-centric User Interaction (chapter 3) discusses these three areas and introduces according models for *Personalization*, for *Ambient Awareness*, and for *Generic User Interaction*. These individual models define the necessary concepts and functional elements in order to meet the requirements of each area. Additionally, a framework is introduced, which describes how these models can be realized coherently. This framework is designated as the *Service Adaptation Framework.*

The approach to realize I-centric User Interaction corresponds to the reference model for I-centric Communications (introduced in section 2.3). According to the reference model, the individual communication space is accessed by using the ambient awareness, personalization, and adaptability features. The three models, identified above, support exactly these features with respect to the I-centric User Interaction aspect. As shown in Figure 2-10, the Service Adaptation Framework depicts the service platform of the reference model. The access network depicts the communication subsystem and various kinds of network technologies providing connections to devices and terminals.

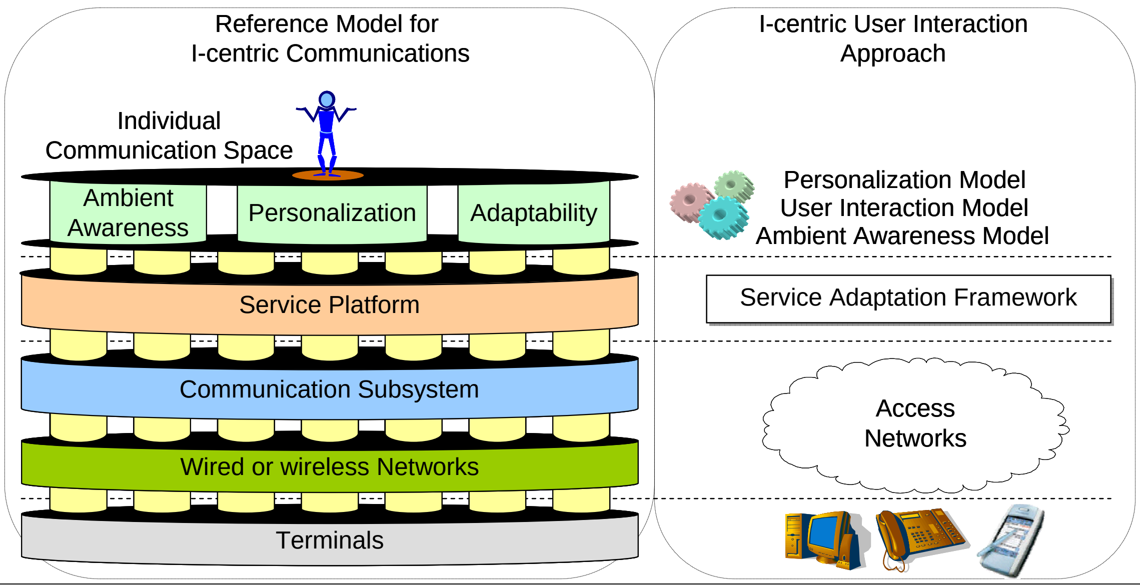


Figure 2-10: The Approach embedded in the Reference Model for I-centric Communications

The model for Generic User Interaction has to reflect the heterogeneity of terminals and communication services. As stated above, this model should not introduce another new technology to realize user interfaces. Instead, the objective is to develop an approach, which supports existing user interface technologies and which is based on relevant standards. For example, the *Device Independence* approach from W3C [W3CDIPRINC] introduces general concepts, which shall be taken into account by the model for Generic User Interaction. These concepts have to be extended to reflect the requirements derived from the vision of I-centric communication.

The following sections introduce the starting points for the development of the presented approach and analyze in detail the objectives of I-centric User Interaction including their meaning in terms of required functions. Specific requirements are identified, which were considered for the development of suitable models to enable I-centric User Interaction.

2.5.1 Personalized User Interaction

The vision of I-centric communication identifies *personalization* as necessary feature to be supported by the I-centric services. As stated in 2.3.2, the objects available in an individual communication space have to be tailored to the preferences of individual users. This also relates to the interaction with these objects. In this way, the preferences of an individual describe the characteristics of the respective communication space including the appearance of the user interface to these objects. Based on the general assumptions of the vision of I-centric Communications on personalization, the following sections identify and analyze more detailed the required functions with special regard to the user interaction.

The term personalization describes the adaptation of the service behavior and the adaptation of the service appearance to user preferences. This means the user preferences have to be gathered and to be considered in the execution of the service and in the realization of the interaction. Users have different demands on the appearance on user interfaces, because of their different capabilities and affections. For example, usually they prefer their native language for the interaction and they possibly prefer different presentation schemas for the optical presentation. To a certain degree, a number of today’s existing software applications support this. The user can choose the language in the installation process and select an appearance model called skins, e.g. the Mozilla WWW browser application [WWWMOZILLA].

In order to adapt to the user preferences, according information has to be gathered from the user. This information describing the user preferences is called personalization data. A competent Personalization model has to provide a suitable information model to structure the personalization data. This information model has to reflect a number of requirements, such as:

- **Distributed storage and exchange with synchronization features:** User preferences can be located on terminals, but should be maintained centrally in the system, likewise. Therefore, corresponding protocols and exchange formats are required. The distributed storage should be transparent to the services, which means that the underlying infrastructure implements the synchronization and provides the valid preferences data to the services. Because, there are and there will be different solutions to enable the personalization of services, the exchange of personalization data should be possible across technology domains, possibly by suitable management functions.
- **Assignment of conditions:** As stated above, the user preferences are usually specific to certain situations, locations, or devices. Therefore, the Personalization model has to provide mechanisms, which enable the user to specify conditions for the selection of preferences, denoted as the *Selection Context*. This should be realized transparent to the users hiding the complexity of the selection algorithm. The conditions should comprise timely specifications, context dependencies including location, and terminal dependencies.
- **Usage of default settings for preferences, which are not specified by the user:** The services should not rely on the existence of specific preference settings for each user. Instead, they should provide a default behavior, if no preferences are specified. This means, services should consider the preference settings, if these are available, but should work based on default settings, if no specific preferences are specified.
- **General and service specific preferences:** If services support the personalized behavior, they have to specify a number of attributes, which can be set by the user. These are the *service preferences*. They are specific to the respective type of services and their meaning is only known by the particular service type. Furthermore, there are preferences, which can be relevant to all kinds of services. These are described as the *general preferences*. Examples for general preferences are languages and presentation schema. Whereas the services have to support the gathering of their service preferences, i.e. within the service interaction, the general preferences have to be gathered in a common way, independent from specific services.
- **Security and privacy issues:** The personal settings provided by the users in order to personalize the system behavior can comprise sensible data, which the user only wants to provide to the services but not to other external entities. The personalization system has to ensure the correct usage and has to prevent any kind of possible misusage. This requires the secure authentication of the component requesting preferences data from the personalization system and according control of the access. In this way, the services are only permitted to access the data, which belong to the respective service, i.e. the respective service preferences, as well as the general user preferences. Beside the access control, the transmission of the data has to be secured by appropriate mechanisms in order to prevent interception and modification.

In addition to the information model to describe preferences, the Personalization model has to provide a functional specification including the definition of required interfaces to access personalization data. The functional specification must fulfill a number of relevant use cases including the gathering, storing, querying, and managing personalization data. On the one hand, the user has to specify the personal settings, which have to be stored and managed as specified by the information model. On the other hand, the services and the user interface components need to access the respective preferences data in order to behave personalized. Furthermore, the Preferences Management System has to provide suitable functions to manage the preferences settings, e.g. to specify the *Selection Context*.

2.5.1.1 Gathering Personalization Data

Personal preferences can be gathered in two different ways: *explicitly* and *implicitly*. The explicit gathering means to interrogate the user by providing suitable user interfaces, in which the user can specify the individual preferences. This approach is not comfortable to the user, because this is time-consuming and not necessarily reasonable for the user. Services should only make a limited use of it. If necessary, these interrogations can be included in the interaction between service and user by providing suitable feedback elements. If no preferences settings are available, first the service behaves in the default way, but can include suitable optional choices in the user interaction. For example, the service can deliver a message to the user, which includes a feedback possibility by which the user can request the service to stop sending messages anymore. This feedback should be evaluated and accordingly stored as new user's preference, which are to be considered in the following service usage.

In this way, the complete user settings can be gathered systematically whenever individual settings are needed for the adaptation. However, some kinds of preference data might be necessary in order to provide a reasonable personalization and therefore require an explicit input, e.g. by providing a suitable input field before the personalization process. The service developers should define a suitable default behavior and should regard the personalization as an additional but optional feature. In this way, the users are not forced to provide the personal settings for the service usage. Nevertheless, they can profit from the personalization feature, if they want.

The implicit personalization is a more suitable approach to realize comfortable personalization systems. It requires surveying and monitoring the user's actions without forcing the user to provide explicit statements. This means, the past service usage should be analyzed to derive individual preferences. The service behavior and the interaction should be designed accordingly in order to draw suitable conclusions. For example, if the user requests a certain type of information while being in the office and request different kinds of information while being out of office, the service can recognize this and adapt its behavior accordingly. Additionally, the conclusions can be drawn by evaluating the behavior of similar users. For instance, certain kinds of user groups, such as people of same life stage or with same profession, have similar interest and preferences. Knowing the preferences of a number of users, the default preferences settings can be adopted similarly for new users, which belong to the specific group.

However, the implicit gathering of preferences and the corresponding adaptation should be transparent to the user. The success of this approach can fail easily, if the user cannot understand the changes. Additionally, the users should always have the possibility to inspect and to change the automatic settings.

2.5.1.2 Storage of Personalization Data

Whereas the services should be provided with a single interface to obtain the user's personal settings, the actual preferences data can reside on different locations. Today, terminal devices, such as personal digital assistants and mobile phones, are able to store user settings. Usually, these devices already provide a certain support for personalization, such as personal ring tones and profiles in today's mobile phones. This means, personalization data can be stored in a distributed manner.

In order to provide the relevant preference data to the services, according mechanisms for the exchange and the synchronization are required. The services should not be forced to retrieve the individual data from different sources. That is why a suitable mechanism for the decentralized storage is needed. However, the personalization system should not rely on the storage of external devices, because the respective personalization data could be lost. Instead, there should always be a copy in the central personalization system. The according synchronization between these components has to be realized in a suitable manner in order not to waste network resources and to provide an efficient storage mechanism. There is also the issue of scalability, which has to be taken into account, if the amount and the size of preference data are extensive.

2.5.1.3 Evaluation of Personalization Data

As stated above, the user should be able to specify conditions for the preferences data, i.e. the Selection Context. Accordingly, the Preferences Management System has to provide a function to evaluate these conditions in order to obtain the valid data and to provide these to the services.

According to the conditions, the Preferences Management System has to obtain the respective data, which are needed to evaluate the conditions. For example, preferences could be bound on a location. This means that these preferences should be used only, if the user resides in this location. In this case, the Preferences Management System needs to receive the user's current position in order to determine the valid preferences set. This evaluation is denoted as the *automatic selection*.

Beside the automatic selection, the user should be able to select preferences settings explicitly. Although there could be preferences settings with a currently valid Selection Context, the user might want to use the service with different settings. In this case, the user should not be forced to change the settings in the Personalization model. Instead, the user should be able to select the preferred settings manually during or before the service usage. This is called the *explicit selection*.

Accordingly, the personalization system has to provide functions to select and to deselect individual sets of preferences. The explicit selection should overwrite the automatic selection. If there is an explicitly selected setting, the evaluation function should return the according preferences set without any further evaluation. If there is no explicitly selected set, the evaluation function should evaluate the specified Selection Context of all available preferences sets and should return the currently valid one. If there is no preference set with a valid Selection Context, the according default settings of the user should be returned instead.

2.5.1.4 Provisioning of Personalization Data

In order to appear personalized, services as well as the user interaction system have to obtain the preferences data from the Preferences Management System. This requires an access to this system to allow querying the relevant preferences data. The access has carefully inspected in order to prevent any kind of misusage. Services have to be restricted to access and modify their own service preferences only. This requires a reliable identification and authentication of the clients accessing the interface of the Preferences Management System. Based on the successful authentication, the actual access should be authorized by appropriate authorization mechanisms, e.g. access control lists.

For the exchange of preference data, a suitable description format is needed. This exchange format should not reflect the complex internal structure. Instead, it should be a lean but flexible format enabling the easy integration into the service implementation. Additionally, the exchange format should be independent from specific platforms allowing the exchange of data among different kinds of systems. This is necessary to enable service providers to develop and to operate their services on a different platform than that of the Preferences Management System.

2.5.2 Ambient aware User Interaction

The vision of I-centric User Interaction identifies *ambient awareness* as a necessary feature to be supported by I-centric services. These services are tailored to the contexts of the individual communication space. This means that the services automatically adapt themselves to the changes of the individual's situation. This adaptation affects the service behavior but also the interaction between the individual and the objects in the communication space. In the following, the aspects relevant to the ambient aware user interaction and the corresponding requirements to the model for Ambient Awareness are briefly presented.

The interaction of the user with a service takes always place in a certain context. The context comprises the environment of a user, which can be described by ambient information. This information can comprise the location, available devices, presence of other people, but also environmental conditions, such as temperature, loudness, etc. It is a very natural characteristic in the human-human communication to consider such ambient information in the communication, which leads to a suitable adaptation of the interaction.

This feature can be applied in the human machine system communication similarly to increase its quality. In this way, the most suitable communication methods and terminals could be chosen and according parameters, such as volume, can be adapted in order to communicate with the user in an adaptive and comfortable way. According to the available ambient information, the presentation of the service has to be adapted.

Adapting the service behavior to ambient information can meaningfully improve the quality of the service and in turn the service experience of the users. The services can adapt more reasonably to the user's current situation, e.g. providing location-based information without interrogating the user about the respective location. In addition, the user interaction can profit from the consideration of ambient information. In this way, the most suitable terminal available in the surrounding of the user can be chosen for the interaction. Likewise, ambient information may allow conclusions whether the user possibly does not want to be disturbed, e.g. because the user is currently in a conversation.

In the I-centric User Interaction, the interaction of users and services should be made comfortable for the user by following the user-centered approach. This means, the characteristic of the ambient aware communication should be supported at best means. Ambient information can be gathered by according sensor devices, which can be also located in the user's terminal. Gathering ambient information means to sense certain physical values, the raw data, which than can be interpreted and revaluated to a more reasonable meaning.

Therefore, the I-centric User Interaction has to provide concepts and mechanisms to gather and to manage ambient information. These should include a suitable information model to describe and to structure the ambient information as well as a number of required functions for the coverage, evaluation, and provisioning. According interfaces to the ambient information management system have to be specified. These should allow the storage of ambient information by the sensor sources and the retrieval of ambient information by the service and the user interaction system.

2.5.3 Delivery Context

According to the previous sections, there is a number of information relevant for the realization of the interaction between user and services. This information comprises the capabilities of the Access Mechanism, the user preferences, and the ambient information. Altogether, they can be described by the term *Delivery Context*. In this sense, the Delivery Context contains all data relevant to the I-centric User Interaction. Furthermore, the Delivery Context has to be provided to the corresponding components, which implement the Service Adaptation, i.e. to the services and to the adaptation of the service's user interface.

Therefore, the concept of I-centric User Interaction has to specify a model and a corresponding description language to describe Delivery Context data. The individual data are provided by respective subsystems, such as the Access Mechanism, the personalization system, and the Ambient Information System. There should be a common approach to filter and to select the relevant data for a specific interaction session. These relevant data should be provided coherently to the services and to the user interaction system. The required context delivery language should enable the exchange and the facile interpretation of the Delivery Context. In this way, the components, which need the Delivery Context data for a reasonable adaptation, do not need to query the respective components separately. Instead, all data of the Delivery Context are compiled and provided through a single point of access.

2.5.4 Heterogeneity of Terminals and Communication Services

In order to determine the requirements of the concepts to realize I-centric User Interaction independent from specific user interface technologies, the existing and relevant technologies have to be analyzed first. In the recent years, there has been a proliferation of types of devices and service Access Mechanisms. These types of devices range from tablet PCs to mobile devices like phones and Personal Digital Assistants (PDA). Connectivity capabilities have also evolved to include high bandwidth modems, LANs, and high-capacity wireless networks (wireless LAN and 3G mobile telecommunication networks). The term *Access Mechanism* describes the complete link between user and the service, which offers the user interface. Therefore, it consists of the terminal and the network connection, which can includes several network nodes and segments. In order to realize a suitable adaptation, the complete Access Mechanism has to be taken into account.

The number of information services available via web technology today is immense. The demand for services will grow in the next years and the user's need to access these services regardless of the terminal device is becoming more and more important. The needs and expectations of the user with regard to access, availability, and consumption of WWW content have evolved. Users now expect to get critical information through different Access Mechanisms from different locations and at different times during their day. They expect access to services anytime, anywhere and via any Access Mechanism. For example, users may want to access some service using the PC connected via a cable network when at home, but when out of the house they expect to access the same service with a mobile phone connected via GSM.

An example of a scenario in which access to a WWW-based service may be required via different Access Mechanisms is described in the following:

A 'parking space' service helping to find free parking spaces can be accessed from the PC at home. The user gets a principal idea where to park his car through detailed maps, videos, and the supposed free areas at the certain time he selected. When he is in the car, he wants the same service to tell him the actual free parking spaces via his mobile phone – by a phone call, a text message, or the presentation of an updated map on a PDA or navigation system installed in the car.

This means, the interaction with such objects in the individual communication space should not only be possible via one selected user interface, such as WWW browsers on personal computers and laptops. Instead, the interaction should be possible through an increasingly variety of other kinds of terminals, including interactive television, voice-only telephones, PDAs, and smart phones. This means, a number of communication methods or communication services should be supported. According to five human senses: sense of sight, sense of hearing, sense of touch, sense of taste, and sense of smell as well as according to the input/output technologies available today, these communication methods can be characterized into:

- Optical
- Acoustic
- Tactile

These are the basic communication methods, which are supported today by technological implementations (as shown in Table 2-1) and which are considered in this thesis. The sense of smell and the sense of taste do not have a practical relevance today, because related technologies are not yet available. All known communication media can be categorized to one of the three identified methods or to a combination of these methods. Besides the heterogeneity of terminals, also these different communication media shall be considered in the provisioning of I-centric User Interaction. The users should be in the position to select the way to interact with a service based on their preferences and current context (available terminals, suitable communication method). For example, while driving a car, the user would prefer a speech-based interaction with a service, whereas being at home, the TV set or a mobile WWW pad could be preferred.

	Optical	***Acoustic***	***Tactile***
Output	text images video	audio speech	Braille device force-feedback joystick and mouse vibrating mobile phones
Input	camera (face and gesture detection)	speech	keyboard mouse

Table 2-1: Media-based Categorization of Communication Methods

Current mobile devices offer mainly single modalities to interact with applications and services. Typical implementations rely on static mechanisms in terms of interface definitions and platform specific deployment. By introducing a structured development process for applications and an interoperation framework where the underlying components are abstracted from the actual interface realization, the ad-hoc creation of multi-modal interfaces for smart mobile devices will be facilitated. A multi-modal interface consisting of various interoperating interface modalities, will support the development of applications that give users a possibility to choose which interface modalities are to be used for content delivery [BUNT]. This will give users a customizable and more natural way of communication.

For example, until mobile phones, PDAs, and SmartPhones appeared on the market, there has been relative little diversity between the devices used to access WWW-based services. Today, there is a huge amount of different types of devices providing access to the WWW. This increasing heterogeneity of the terminal devices and networks leads to new problems for service developers. Providing services for a single terminal type would exclude large numbers of users. [CONVIGO] describes the multiple channels that need to be dealt with when developing services as depicted in Figure 2-11.

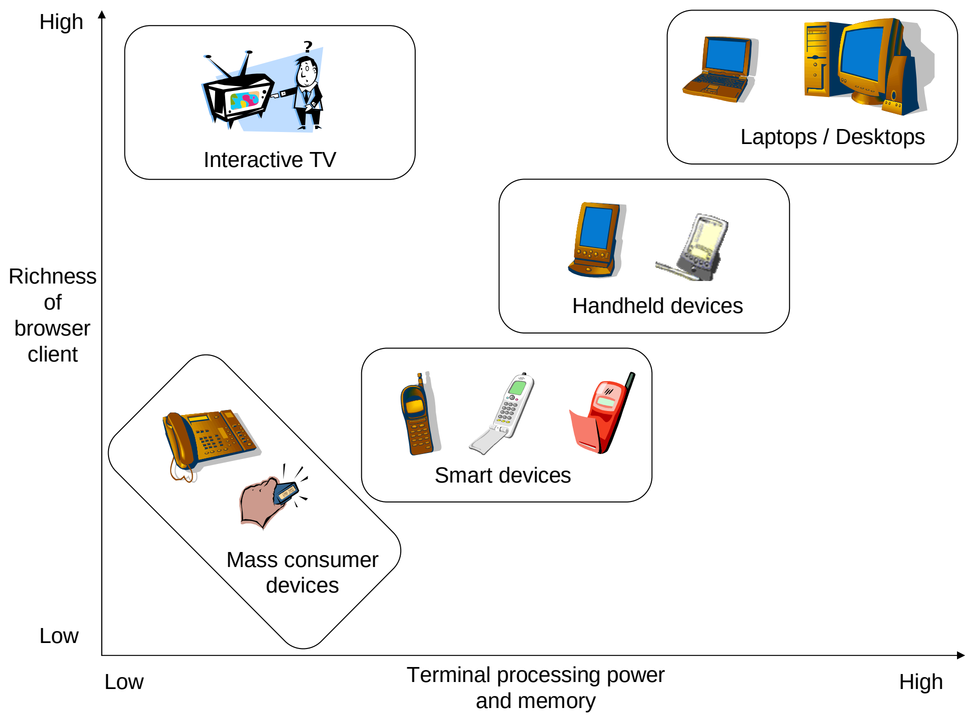


Figure 2-11: Range of Channels and Modes

To offer access to services through heterogeneous terminal devices and network technologies, service providers have to develop their services specifically for many devices capabilities. However, developing content for several individual devices has a number of disadvantages:

- Expensive and time consuming in developing and maintenance
- Fragmentation of information
- Learning process for each new device and communication media
- Excluding users whose preferred device and communication media is not supported
- Excluding future terminal devices and networks

This shows that re-authoring the content and the service's user interface in order to support other user interface technologies and to reflect the different capabilities of each device is clearly impractical. The development of device dependent services depicts an inappropriate way to support numerous terminal devices and networks. It is the key challenge for service authors to enable their services to be delivered through a variety of Access Mechanisms with a minimum of effort.

2.5.4.1 Characteristics of Terminal Devices

Today, there is a multitude of different types of terminal devices, which support different communication media. Among these devices, there are huge variations in capability of processing, display, and input. Because service developers cannot be aware of all characteristics of existing and future devices and networks, authoring techniques must enable them to support the diversity of devices. Some characteristics of the Access Mechanism are closely associated with the device as describe by following paragraphs.

Screen Size and Resolution: While the physical size of the screen is a static property, the resolution actually in use may vary. The resolution can be changed by the user, and the area available to the user agent running on the device, e.g. a WWW browser, may be different. Workstations sometimes use monitors to act as a window onto a larger desktop. On smaller devices, the resolution changes are less common, but devices such as PDAs may use displays in portrait or landscape mode. Screen size and resolution are important for Device Independence because they influence the size of media elements that can be displayed and the layout used for the presentation. User interfaces might be very limited in the content and number of elements that can be displayed at once. Authors may need to design separate layouts for different resolutions and to change the organization of the output.

Color Capability: This is usually a static property. It is described by numbers like a certain color depth of the screen or a gamma factor. Like the resolution, it affects the media elements that can be displayed.

Video Capability: Some devices themselves or the user agents running on them are able to display video clips. Only media elements can be played back, which have the proper format (coding) suitable to the respective device.

Audio Capability: Similar to the video capability, some devices are inherently capable of playing audio clips. Video and audio often form a unit. Audio capabilities can be considered static, too, once they are available on a device. As with the characteristics already mentioned, it is important for device independent presentations to take notice of the audio capabilities when media elements are selected.

Input Capabilities: On the one hand, workstations equipped with full keyboard are convenient for input of large quantities of data. On the other hand, mobile phones offering a simple keypad for input are generally not convenient. The ease of use of a device's input facilities is of importance for the service author. Some interactions need to be simplified or omitted for devices that do not offer comfortable input facilities. For example, complex application procedures may simply be inappropriate for mobile phones. Authoring techniques must enable the service author to control how interactions are expressed on the target devices.

2.5.4.2 Network Characteristics and available QoS

In addition to terminal specific capabilities, also the characteristics of the network connection have to be considered for the adaptation of the user interaction. For example, a powerful state-of-the-art laptop is able to play back a high-resolution video stream. However, being connected via wireless link, such as GPRS, it is not reasonable to deliver the video without any adaptation. Therefore, the characteristics of the network connection used in a communication session have to be included in the adaptation process. The following characteristics are relevant:

Declared and Actual Bandwidth: The declared bandwidth[7] is the available speed of data transfer. It is a static property while the actual bandwidth may vary from minute to minute depending on many factors, e.g. how busy the network is or the distance of a mobile phone from the base station. The declared bandwidth, e.g. 9600 bps with a GSM phone, is a good starting point for adaptation decisions, but available bandwidth has to be determined from time to time. The bandwidth is an important consideration for device independent authoring techniques. Service authors may want to choose from a number of available alternative media elements appropriate for the bandwidth. For example, they might want to deliver a video clip when high bandwidth is available instead a still image.

Delay and Jitter: The delay is an important factor for the latency when using a service. Services that have to be used with real time interactions are probable not applicable on networks with a high round-trip time, e.g. a mobile phone connected with GPRS. The delay time is more interesting for the service itself, not for the presentation. Whereas, the delay describes the time the information exchange needs from the transmitter to the receiver, the attribute jitter describes the temporal variation of the delay.

Costs: Users are usually charged for the use of network communication services. The service providers use different pricing models for the particular services (e.g. GPRS vs. ISDN dial-up). These models differ also from provider to provider. The parameter 'costs' refers to these transmission costs. From the user's perspective, it is also important to consider, what kinds of costs occur in the interaction. Whereas, using a low-priced connection the user might accept extensive content, but using a very expensive connection, he surely would insist on retrieving only the minimum necessary content.

2.5.4.3 Multitude of User Interface Technologies

The interaction between human being and the objects in the individual communication space requires a *user interface*, which mediates between human and the technical peculiarities of these objects. The I-centric User Interaction addresses this user interface in particular providing the required adaptation capabilities. The specification and the development of these adaptation capabilities require analyzing briefly the different aspects of relevant user interface technologies.

A *user interface* facilitates the interaction between a computer system and a human being (user). It transforms computational information into signals humans can perceive with their senses and, vice versa, transforms the user's input into computational signals.

Specific user interface technologies can be further characterized to support the output and/or the input of information. Output of information means to present the information to the user, whereas the input of information means to pass the information given by the user to the service logic (application).

Because of technical limitations, the human-machine interaction nowadays still concentrates mainly on optical information delivery (graphical user interfaces) and tactile user inputs (key-

[7] Note that the term 'bandwidth' is used within this thesis synonymously to 'bit-rate' as it became common in computer science, in contrast to the usage as a range of frequencies in electrical engineering.

boards, mice, or touch screens). The meaning of speech-based communication increases with the improvement of necessary technologies – speech synthesis and speech recognition. Speech is the human's natural way to communicate. Therefore, speech-based human-machine interfaces are expected to play an important role in the future. Apart from special purpose applications (e.g. Braille devices), other senses like tasting and smelling do not have much practical relevance today in human machine communication, although according solutions and concepts are being researched intensively.

The employment of human senses in human-machine communication depends on the ability of generating suitable signals and interpreting human signals by machines. The understanding and imitating of the human's way to communicate is one of the biggest challenges in computer science. Therefore, today human-machine interfaces concentrate on graphical presentation of information sometimes accompanied by simple sounds with tactile input devices, e.g. keyboards and pointing devices.

Throughout history of computer science, people have raised the level of abstraction with which they communicate with computers. Originally, people programmed computers in binary machine code. Later, assembly language was a big revolution: people could write programs using mnemonics instead of strings of zeros and ones. Then programming languages and compilers came. Programmers resisted high-level programming languages at first because compilers generated less efficient machine code than hand-coded assembly programs. However, high-level programming languages allowed a wider range of people to program.

The advent of the WWW again raised the abstraction level again. For the first time, documents could be published by anyone in a platform-independent format. The Hypertext Markup Language (HTML) version 2 empowered non-programmers to create simple forms-based interfaces. Dynamic HTML (DHTML) added some direct manipulation mechanisms in order to improve the flexibility of the design and to enable unhesitating interaction with the user. Style sheets added further abstraction by separating content and presentation style, simplifying porting and customization. Because DHTML requires a complex interpretation environment to render the interface, new markup languages were proposed for small appliances: the wireless Markup Language (WML)[8] and the Compact HTML (cHTML)[9].

However, each of these languages embeds specific assumptions about the type of interface or appliance on which they are used. HTML describes a document, WML describes cards for a handheld appliance with small screen, VoiceXML assumes voice, and so on. User interface technologies are continuously advancing. Accordingly, plain graphical user interfaces with pointing devices are increasingly supplemented with natural sounding speech synthesis, voice and handwriting recognition, full motion video, virtual reality and even mechanisms to receive input from brain waves. These developments argue for one more step in abstracting user interfaces – good for both today's and future user interfaces.

2.5.4.4 Different Communication Modes

There are different modes of communication. These modes reflect requirements given by the kind of the information and given by the context, in which the communication takes place.

On the one hand, the following communication modes can be identified:

- *Point-to-point* communication (a conversation in private) – two peers communicating with each other
- *Point-to-multipoint* communication (a discourse given before an audience) – one communication peer provides information to a group of others

[8] [WAPWML20]

[9] [IMODE]

- *Multipoint-to-multipoint* communication (a discussion in a group with more than two people) – a group of communication peers exchanging information, each peer can be sender and receiver
- Broadcast communication (television and radio broadcast) – information is sent to a group without knowing who is receiving the information

On the other hand, communication can be:

- *Synchronous*, where both parties are involved at a time
- *Asynchronous*, where only one party is involved at a time

Synchronous communication services are real-time dependent. A fixed time frame exists for transmitting a part of the information from the sender to the receiver (end-to-end delay). If the time frame is exceeded the synchronous communication will be broken. A subclass of synchronous communication is isochronous communication, with constraints to the jitter.

Asynchronous communication services do not depend on any time conditions, therefore best effort strategies can be applied. For example, the store and forward message handling for e-mail works on these principles. Table 2-2 gives an overview of relevant telecommunication services:

Asynchronous Telecommunication Services		
Service	***Terminal***	***Network***
Electronic Mail	Computer	Internet
Instant Messaging	Computer	Internet
SMS (EMS)	Mobile phone	GSM
MMS	Mobile phone	GSM/Internet
Pager	Pager device	Wireless network
Facsimile	Fax	PSTN / GSM
Synchronous Telecommunication Services		
Service	***Terminal***	***Network***
Telephony	Telephone / mobile phone	PSTN / GSM
Internet telephony (VoIP)	Computer / telephone	Internet

Table 2-2: Asynchronous and Synchronous Telecommunication Services

2.5.5 Device independent User Interaction

The currently existing as well the future user interface technologies differ in supported media, interaction schema, required capabilities, and are suitable to different situation and application scenarios. The service developer has carefully to select the most suitable user interface technology for the service in order to increase the potential customer base. To support more than one user interface technology is still complicated and requires extensive competencies. For example, traditional banking services were originally exclusively provided in the different branches, but were then extended by banking terminals, online banking applications in the Internet, speech-based telephony banking services, as well as WAP and SMS based online banking services in the mobile telecommunication networks. These different kinds enable the user to access the same banking functions at any time, from different terminals, and from different locations. The users profit from the achieved flexibility, which in turn leads to a stronger market position of the respective company. The general approach is depicted in Figure 2-12, taken from [VEDATI].

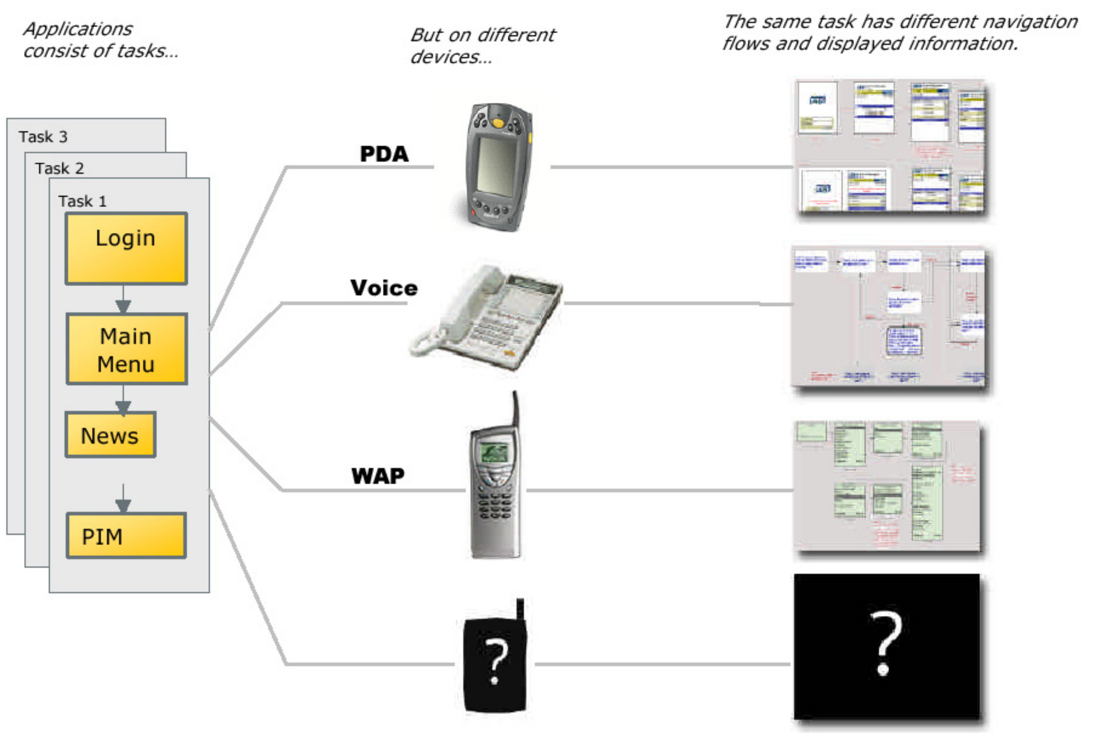

Applications consist of tasks...
But on different devices...
The same task has different navigation flows and displayed information.
Task 3
Task 2
Task 1
Login
Main Menu
News
PIM
PDA
Voice
WAP
?
?

Figure 2-12: Single Application, Multiple User Experiences

However, there is no common solution to realize a service and to provide this service with a number of different Access Mechanisms. Today, these applications are realized with a specific support for all supported user interfaces. On the other hand, there are a number of research activities, which tackle this problem by developing suitable approach for a *device independent user interaction*, such as the Single Authoring approach from the W3C [WWWW3C]. For example, this approach defines the *Device Independence* principle [W3CDIPRINC], which aims for the authoring, generation, and adaptation of WWW content and applications for the interaction via many different Access Mechanisms to the WWW. Suitable approaches to realize the Device Independence principle are introduced in [GOEBEL, GOEBEL2].

The W3C approach is restricted mainly to WWW-based applications, but the general idea fits to the vision of I-centric User Interaction. The I-centric User Interaction shall enable a device independent realization of the user interaction, denoted as *Generic User Interaction*, by separating the user interface from the actual service logic. The separation requires a suitable *Generic User Interaction Markup Language*, which is used by the services to describe the interaction, i.e. the dialog and the content, which is then to be transformed to the presentation of the respective user interface technology. In this way, the services provide a *generic user interface* and do not need to implement the support of specific user interface technologies. Hence, they can profit from the wide range of existing user interface technologies to provide a comfortable access to the service.

Beside the actual specifications and concepts of the different user interface technologies, the transformation has to consider the specific capabilities of the Access Mechanism. The Access Mechanism describes the connection of the user's terminal through an access network to the service. On the one hand, the capabilities of the individual terminals differ, for instance the screen size of desktop computers, personal digital assistants, or mobile phones. On the other hand, the network connections have fluctuating capabilities in terms of transfer rate, delay, or jitter. All these characteristics have to be taken into account for the automatic transformation of the generic user interface to the real user interface to be presented on the user's terminal.

Therefore, the I-centric User Interaction approach has to provide a concept for the generic description of user interfaces and for the adaptation of such generic user interfaces according to the respective capabilities of the Access Mechanism. The concept should not be limited to particular user interface technologies of today’s terminal devices, but should be open and flexible to add technologies of tomorrow’s terminal devices, likewise.

Beside the capabilities of the Access Mechanism, the adaptation of the generic user interfaces has also to consider the user’s preferences, which leads to a personalized appearance of the user interface and a personalized user interaction. Usually, users have different preferences for the presentation of a service, similarly to the preferences for the service behavior. The user interaction should be realized in an intuitive, comfortable, and user-friendly way. The user must not recognize the automatic creation of the user interface.

2.6 Summary

This chapter presented the general vision of I-centric Communications and introduced the aspect of I-centric User Interaction. Two different facets of the interaction between the individual and the objects in the individual communication space are identified: the *implicit* and the *explicit* user interaction. The concept of I-centric User Interaction concentrates on the explicit interaction facet. According to this vision, it requires that the interaction is adapted to the capabilities of the Access Mechanism, to the user’s preferences, as well as to the user’s current situation.

Based on the first analysis of this adaptation demand, specific functions and relevant requirements are identified. These cover the support of a wide range of different user interface technologies as well as different kinds of terminal devices. In this way, the demand to support the users in the interaction with the different objects in the individual communication space in a flexible and easy-to-use manner can be fulfilled. In order to improve the user interaction furthermore, the demand to support the personalization and the ambient awareness of the user interaction is identified. These capabilities complete the attempt to imitate the characteristics of human-human interaction at best means.

Summarizing, the I-centric User Interaction approach requires solutions for:

- A coherent Service Adaptation Framework implementing the different functions and providing suitable interfaces to the services
- The personalized user interaction
- The ambient aware user interaction
- The device independent user interaction
- The management of the Delivery Context containing all data necessary for a suitable adaptation

These general requirements describe the relevant areas of concern and depict the starting points for the development of an approach to enable I-centric User Interaction. As identified, the approach to realize I-centric User Interaction consists of a Service Adaptation Framework as well as suitable models for Personalization, for Ambient Awareness, and for Generic User Interaction focusing on the different areas of concern. Based on the identified objectives and requirements, the following chapter 3 introduces the complete approach in detail.

CHAPTER 3 Approach to Realize I-centric User Interaction

This chapter introduces an approach for I-centric User Interaction, which enables users to interact with the objects in their communication space in a flexible and comfortable way. Following the I-centric vision, users are supported to use their favorite communication method: speech, graphical, synchronous, asynchronous, etc., suitable to their active context and the type of intended action. Regardless of the applied user interface technology, the interaction shall adapt to user's preferences and current situation. Accordingly, this chapter introduces a Service Adaptation Framework, which specifies the necessary functional elements to realize I-centric User Interaction. The functional elements are introduced detailed by the models for Personalization, for Ambient-Awareness, and for Generic User Interaction.

3.1 Introduction

I-centric User Interaction enables the flexible and comfortable interaction between individuals and the objects in their individual communication space. As depicted in Figure 3-1, the I-centric User Interaction approach realizes a kind of a *user interface*, which surrounds the user, appears personalized, and adapts to the respective situation. The user does not have to adapt to the individual technical peculiarities of the different types of objects, which might consist in the communication space. Instead, the I-centric user interface provides access to these objects, i.e. the interaction with these objects, in a flexible and comfortable way.

According to the vision of I-centric communication, I-centric services support the ambient awareness, personalization, and adaptability features. Similarly, the interaction between individual and objects should be characterized by these features as well. Therefore, the approach to realize I-centric User Interaction has to provide suitable concepts to fulfill these demands.

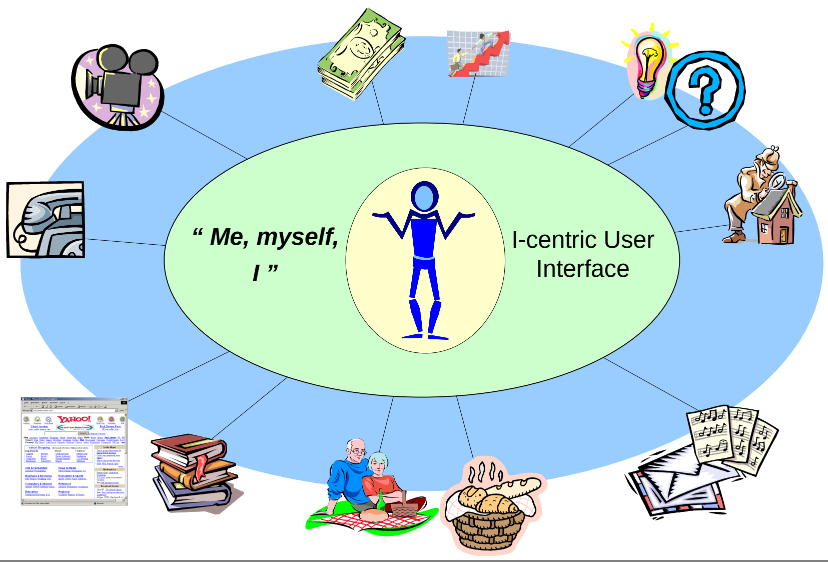


Figure 3-1: The I-centric User Interface

The objectives of I-centric User Interaction and the derived requirements, which are described in section 2.5, identify three different areas of concern regarding the adaptation of the user interaction to:

- The user's preferences
- The current environment of the user
- The particular terminal and network used in the interaction

These areas of concern are interconnected and have to be considered coherently. The approach to realize I-centric User Interaction discusses these three areas individually and introduces according models for each. These models comprise:

- A model for *Personalization* – supporting the management of user preferences (section 3.3)
- A model for *Ambient Awareness* – supporting the management of ambient information (section 3.4)
- A model for *Generic User Interaction* – supporting the realization of user interfaces independent from specific user interface technologies and therefore supporting heterogeneous terminals and media (section 3.5)

These individual models define the necessary basic concepts and functional elements in order to meet the requirements of each area of concern. Because of their interconnection, the individual models partly refer to the respective other models. Based on these models, a framework has been developed, which describes how these models can be realized coherently. This framework, designated as the *Service Adaptation Framework*, is introduced in section 3.2 giving also a brief overview about the elements and concepts of the three models. The individual models are presented in detail in separated sections (3.3-3.5) after the introduction of the Service Adaptation Framework.

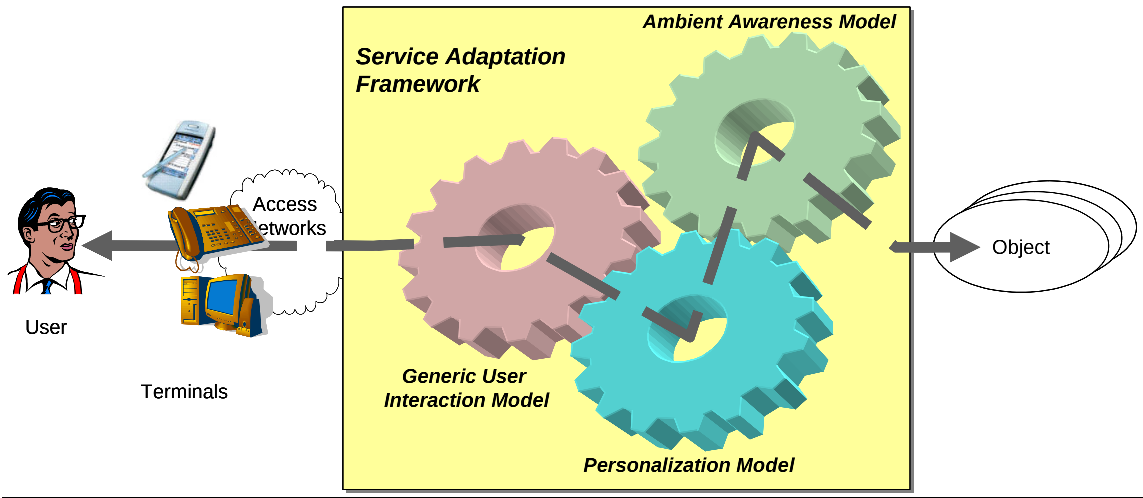


Figure 3-2: The I-centric User Interaction Models and the Service Adaptation Framework

As depicted in Figure 3-2, the Service Adaptation Framework implements supporting functions for all three models. The interaction between user and objects is adapted by utilizing the functions provided by the framework. In this way, objects do not need to implement the complete models themselves. By using the framework supporting the identified models, they can profit from the corresponding concepts and improve their interaction with the user according to the vision of I-centric User Interaction.

Figure 3-3 describes the general relationships of the main functional elements of an I-centric User Interaction system. There is the *object*, which offers the user interface supporting the Generic User Interaction model. According to the requirements, the user can interact with the service using any kind of communication service and any kind of terminal. The *Service Adaptation Function* translates the generic user interface, provided by the service, into the specific user interaction technology and adapts it to the particular capabilities of the terminal and to the characteristics of the access network. As shown, the adaptation of the user interface takes also the *user's preferences* as well as the available *ambient information* into account.

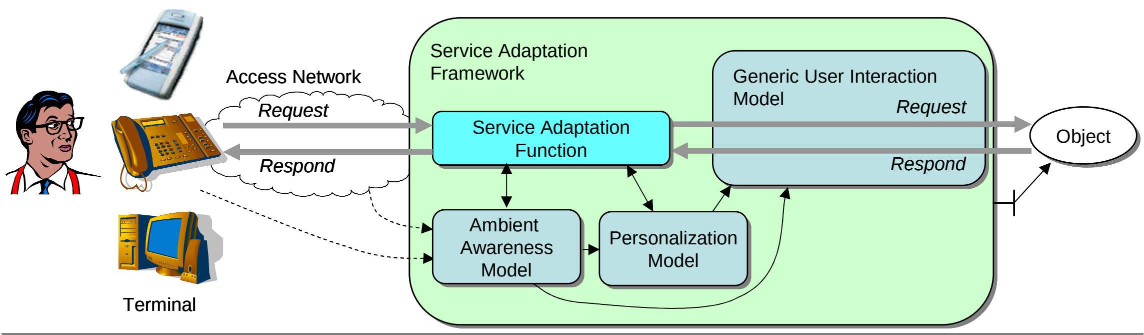


Figure 3-3: Relevant Components for I-centric User Interaction Models

To give a coarse understanding, the following sections briefly give an initial overview of the Service Adaptation Framework and the three models. Because of their interconnection, a general understanding of the goals and approaches of each is necessary in order to follow the detailed introduction.

3.1.1 Service Adaptation Framework

The *Service Adaptation Framework* (section 3.2) contains the necessary functional elements to realize the three different models for Personalization, Ambient Awareness, and Generic User Interaction. In this way, it provides according support and interfaces to the services. Corresponding to their scope, the models define general concepts, information models, protocols, and corresponding APIs. The specification of the Service Adaptation Framework is based on these models and defines their relationships as well as their coherently realization detailed. From the viewpoint of service developers, the Service Adaptation Framework provides the interfaces to the respective functions enabling Personalization, Ambient Awareness, and Generic User Interaction.

3.1.2 Personalization Model

The *Personalization* model (section 3.3) describes concepts for the management of user preferences. In order to appear personalized, services and user interfaces have to consider the preferences of individual users or groups of users, i.e. they should adapt to them in the best possible way. This model contains an information model to structure preference data, defines a programming interface and the interaction schemas for the usage of this information by other components, such as the services and the Service Adaptation Function.

Whereas the Personalization model itself concentrates on the managements and provisioning of preference data, the actual process of personalization has to be implemented by those components, which realize the services and the interaction with the user. Because personal preferences can be related to situations, the Personalization model considers ambient information provided by the Ambient Awareness model.

3.1.3 Ambient Awareness Model

The *Ambient Awareness* model (section 3.4) describes concepts and mechanisms to gather and to manage ambient information (describing for example the user's environment comprising location data, available terminals and devices, climate data, etc.) With such data, services can adapt more reasonably to the user's current situation, e.g. providing location-based information without explicitly interrogating the user. In addition, the user interface can profit from the consideration of ambient information. In this way, the most suitable terminal, available in the surrounding of the user, can be chosen for the interaction.

For example, ambient information can also allow to draw conclusions whether the user possibly does not want to be disturbed, e.g. because the user is in a private conversation. The Ambient Awareness model defines an information model to describe and to structure ambient information as well specifies necessary interfaces for the coverage, evaluation, and provisioning of ambient information.

3.1.4 Generic User Interaction Model

The model for *Generic User Interaction* (section 3.5) enables the device independent specification of a user interaction. The model separates the user interface from the service logic by defining a markup language, which abstracts from technical details and peculiarities of the different user interface technologies (WWW, WAP, speech, messaging, etc.) Services describe their user interfaces in this language, instead of implementing the specific user interface technologies directly. This depicts a separation of the user interface and the service logic. In this way, service developers do not have to know these technologies in detail and do not have to spend necessary development resources on their integration.

Furthermore, this model defines a Service Adaptation Function, which understands the model's markup language and can transform it to specific user interface technologies. In this way, a service does not have to be aware of the specific user interface technology and the concrete terminal, which is used in the interaction. For example, the user can use a telephone with a speech-based user interface, WWW, or WAP-based access. The generic user interface, provided by the service, is transformed automatically to the according technical presentation as required by the respective user interface technology. Furthermore, the Service Adaptation Function realizes the user interaction also by applying innovative concepts such as multi-modal user interaction or content preparation applying advanced summarizing capabilities.

In this way, flexible and comfortable user interfaces can be realized supporting all kinds of user interface technologies whereas keeping the effort for the service developer low. The Generic User Interaction model uses also the Personalization model and the Ambient Awareness model to consider relevant user preferences and relevant ambient information in the realization of the user interaction.

3.2 Service Adaptation Framework for I-centric User Interaction

The Service Adaptation Framework, presented in the following sections, was developed based on the identified requirements in order to support I-centric User Interaction. It specifies how the three different models to realize I-centric User Interaction work together realizing a comprehensive platform to support I-centric services. The presented specification identifies the essential functional elements, describes the adaptation process detailed, and introduces the necessary interfaces and formats for data exchange. Details about the functional specification of the different elements are provided in the respective sections, which introduce the individual models entirely after the presentation of the Service Adaptation Framework, i.e. section 3.3-3.5.

3.2.1 Introduction

The approach to realize I-centric User Interaction identifies three areas of concerns and provides corresponding models for each. These models comprise suitable concepts for the *Personalization*, *Ambient Awareness*, and *Generic User Interaction*. The individual models are introduced in the respective sections individually enabling a possibly separated realization of each. However, the objectives of I-centric User Interaction require the coherent implementation of all three models to support a personalized, ambient aware, and device independent user interaction.

The *Service Adaptation Framework*, which is presented in the following sections, describes an approach to realize a platform for I-centric User Interaction. This platform implements coherently the concepts introduced by the three models. It provides the necessary functions and interfaces to support services in the interaction with the user in an I-centric manner.

The model for Generic User Interaction specifies the separation of user interface and service logic by defining a suitable language to describe the user interaction in a technology independent way. The Service Adaptation Function transforms the generic user interface into the technology specific representation. The transformation process should consider personal preferences as well as ambient information in order to let the interaction appear personalized and ambient aware. The gathering, management, and provisioning of preference data as well as of ambient information are specified by the according models for Personalization and Ambient Awareness.

This means, the Generic User Interaction model depicts the starting point for the development of the complete Service Adaptation Framework. The Personalization model as well as the Ambient Awareness model has to provide the necessary data, relevant for the adaptation process. All these data have to be incorporated into the *Delivery Context* and to be provided to the Service Adaptation Function as well as to the services. Additionally, a suitable session concept is required. This should enable flexible session management supporting the suspension and reactivation of Service Sessions possibly on different terminals, i.e. with different Access Sessions.

The internal complexity of the individual solutions should be hidden from the external component, i.e. the services. Suitable interfaces shall enable an easy and comfortable usage of the functional elements. In addition, relevant security issues have to be considered in the specification and realization of the framework interfaces.

3.2.2 Overview

The *Service Adaptation Framework*[10] depicts a platform to realize I-centric User Interaction by implementing the individual concepts of the three introduced models. The *Service Adaptation Function*, introduced by the Generic User Interaction model, transforms the generic user interface, provided by the service, into the technology specific representation. The transformation process takes relevant ambient information as well as user preferences into account. Figure 3-4 shows a general overview of these relationships.

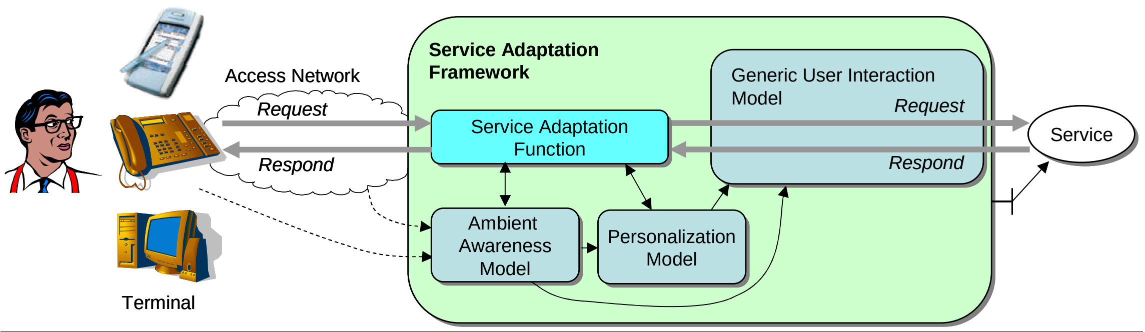


Figure 3-4: Service Adaptation Framework Overview

This means, the Service Adaptation Framework provides the necessary functional elements to implement the identified models to realize I-centric User Interaction. Whereas the individual models provide necessary concepts including specifications of data structures and functions according to their scope, the Service Adaptation Function implements the actual realization and adaptation of the user interface. This function uses the information provided by the functional elements implementing the individual models.

In order to interact with an object of the individual communication space, the individual can use any kind of terminal and communication service, supported by the Service Adaptation Framework. The object has to provide an according user interface specification, which can be transformed and presented to the user by the Service Adaptation Function. Because, abstract objects such as ‘food’ or ‘money’, which can exist in the individual communication space, do not necessarily provide the user interfaces themselves.

Instead, corresponding I-centric services, which manage these abstract objects, realize the user interaction with these objects. In this sense, the I-centric services will provide the generic user interface and map the interaction to the particular functions, which are provides by the objects. In general, the presented approach is not confined to either support the interaction with objects of the individual communication space or with the I-centric services, which manage these objects. Accordingly, the term ‘service’ is used in the following homogenously for the objects, i.e. the services they provide, and for the I-centric services.

As depicted in Figure 3-4, the service describes the user interface in the *Generic User Interaction Markup Language* (GUIML), which is specified by the Generic User Interaction model (section 3.5.4). The Service Adaptation Function translates the GUIML source to the device and communication service specific presentation, e.g. WML, VoiceXML, or HTML. For the adaptation, it considers the respective Delivery Context.

[10] An initial approach of the service adaptation framework was introduced in the diploma thesis [MROHRS].

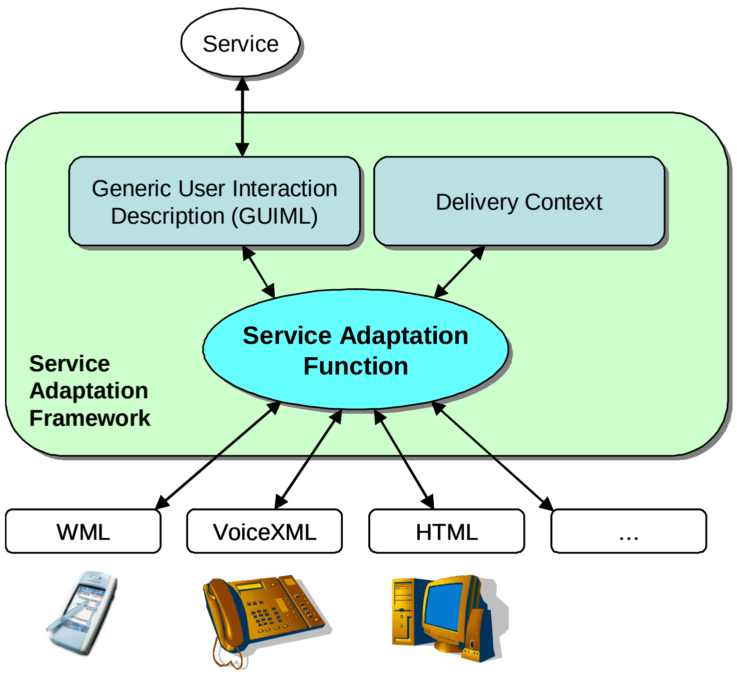

Service
Generic User Interaction Description (GUIML)
Delivery Context
Service Adaptation Function
Service Adaptation Framework
WML
VoiceXML
HTML
...

Figure 3-5: The Service Adaptation Function

In section 3.5.6, the Generic User Interaction model also introduces the *eXtendible Delivery Context Language* (XDCL) to describe the Delivery Context and specifies suitable modules to describe the capabilities of the *Access Mechanism*. The Personalization model and the Ambient Awareness model manage additional data imperative for the personalized and ambient aware appearance of the user interfaces and of the actual service logic. The complete Delivery Context therefore consists of:

- Capabilities of the Access Mechanism
- User preferences
- Ambient information

Using XDCL, all these data can be described coherently as a single data structure.

In order to derive a functional model to realize I-centric User Interaction, the possible usage scenarios have to be analyzed carefully. The usage of a service is manifold depending on the used communication service, i.e. user interface technology. According to the initiation of the service usage, the following two possible aspects can be identified:

- User initiated (pull paradigm)
- Service initiated (push paradigm)

Whereas the user can contact a service by using a terminal and opening a connection to the service, the service can similarly get in contact with the user. Both aspects are relevant to support comprehensive and user-friendly service scenarios. For example, a news service should have the possibility to send messages to the user (push) and do not have to force the user to check regularly for new messages (pull). Only, both mechanisms together can provide the necessary service quality to attract users.

In the user initiated scenario, the used terminal is already known before the actual service usage. This means there is already an Access Session between user and system. In contrast to this, the service initiated scenario requires to establish such Access Session first, i.e. before the actual service content can be delivered to the user. Therefore, a suitable session concept, which reflects and supports both scenarios, is required.

Furthermore, the communication connection between user and service can be characterized in terms of their duration. They can be *synchronous* or *asynchronous*. Synchronous communication sessions have a certain duration as long as the dialog between user and service continues. There is a communication context, which describes the state of the communication. A synchronous communication session lives as long the user or the service do not conclude the communication. On the other hand, an asynchronous communication relationship depicts the single delivery of a message, i.e. service content or user request. The communication starts with the initiation and finishes immediately after the successful delivery. There is no context determining the state of the communication. Both, user initiated as well as service initiated communication sessions, can be synchronous and asynchronous.

The following sections introduce detailed the concepts and the individual functional elements of the Service Adaptation Framework.

3.2.3 Session Model

According to the different usage scenarios and characteristics of the communication connections, a suitable session model is required. This model has to maintain all the data, which are relevant to the ongoing interaction between user and service. Figure 3-6 shows an overview of a suitable session model.

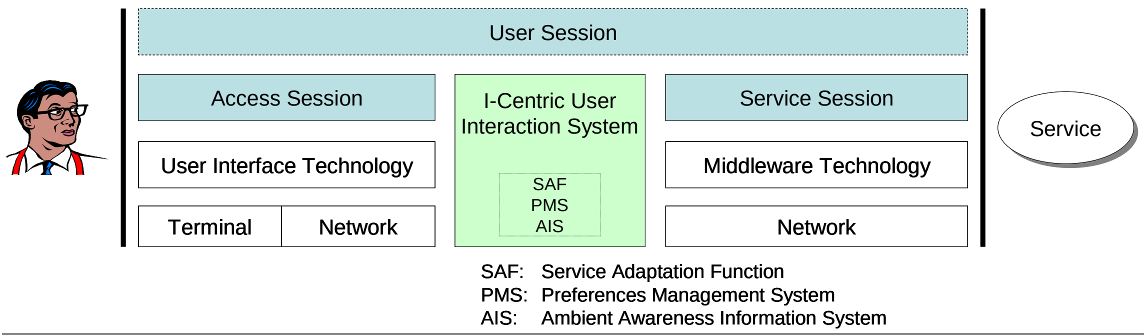

User Session
Access Session
User Interface Technology
Terminal
Network
I-Centric User Interaction System
SAF
PMS
AIS
Service Session
Middleware Technology
Network
Service
SAF: Service Adaptation Function
PMS: Preferences Management System
AIS: Ambient Awareness Information System

Figure 3-6: Session Model for I-Centric User Interaction

The general relationship between user and service is depicted by the *User Session*. This session consists of the *Access Session* and the actual *Service Session*[11]. The Access Session describes the connection of the user's terminal to the I-centric User Interaction system and is therefore very specific to the type of communication service used, e.g. speech, WWW, or WAP. The user can maintain multiple connections with different terminals to the I-centric User Interaction system. Then, there are multiple Access Sessions for each terminal accordingly. When the user is interacting with a service, there is one Service Session describing the state of this interaction dialog. This means, the user can have multiple Access Sessions, which are assigned to the same Service Session. In this way, multi-modal user interaction can be realized. Furthermore, the user can interact with different services, i.e. multiple Service Sessions, through one terminal, i.e. one Access Session.

3.2.3.1 Access Session

According to the nature of the selected communication service, the Access Session is synchronous or asynchronous. Messaging-based communication systems, such as e-mail, SMS, or MMS, are always asynchronous whereas speech-based interaction is synchronous. Because there is an immediate response in WWW and WAP applications usually, they can be regarded as synchronous although their actual implementation uses asynchronous interaction schemas. They also apply session concepts usually to store the state of an interaction between the individual requests.

[11] The session concept defined here was derived from the TINA Service Architecture described in [TINASA].

For asynchronous communication systems, this means that there is actually no need to maintain an Access Session. Because, this session is designated to contain necessary data for the service interaction, the services might rely on its availability. Therefore, an I-centric Communications system should create an according Access Session even for asynchronous services and should provide it to the system components as well as to the particular service. Whenever, the asynchronous communication exchange is finished, i.e. the message was delivered, and the service does not need the respective data anymore, this session can be dissolved. However, the service or the user might respond in the same way as they have received the message, which means that the same Access Session can be used further on. It depends on the actual realization how this is implemented in a real system.

The several user interaction technologies are different in the support of the user initiated and service initiated service usage. The messaging solutions as well as the speech-interaction systems can be triggered by both, the users and the services, whereas the WWW model only supports the user initiated service usage. To trigger a WWW-based access, the service however can deliver an according link, i.e. URL reference, to the user by other communication services, such as by an e-mail. WAP on the other hand supports the service initiated usage through the WAP push mechanism.

The Access Session describes all the characteristics of the Access Mechanism, i.e. the connection of the user's terminal to the I-centric User Interaction system.

3.2.3.2 Service Session

The Service Session describes the current state of a service usage. When the user is interaction with a service, there is a corresponding Service Session. The Service Session refers to one or more user sessions. If there is no user session anymore, the service usage was concluded. This means the respective Service Session is not needed furthermore. However, the system may support the interrupting of an interaction and the later resuming of the interaction at the same state. This requires that the Service Sessions outlive the user sessions by using corresponding *suspend* and *resume* mechanisms.

Basically, it depends on the implementation of the I-centric User Interaction system, whether Service Sessions are realized as persistent sessions, i.e. independent from existing user sessions, or as non-persistent sessions, i.e. dependent on existing user sessions. The option, which model should be selected for a particular service, could be provided to the service providers. This could be specified in the service deployment process then. However, the concrete realization should ensure transparency to the user, i.e. the user should not experience the technology driven details such as being confronted with an option to suspend or to resume a service.

3.2.3.3 Service Interaction Initiation

In a **user initiated interaction**, the Access Session is created first. With the terminal, the user contacts a service through the I-centric User Interaction system. This means the terminal first creates a connection to the I-centric User Interaction system, which in turn creates and maintains an according Access Session for this connection. All characteristics of the terminal connection including the obtained user identification are described by the Access Session.

After the Access Session is created, the system examines if there is a corresponding Service Session for the user, which should be resumed. If not, a new Service Session is created and the service is contacted with the initial request of the user. The service can now access the Service Session, i.e. resumed or newly created, as well as the Access Session and can obtain the necessary data from it. The service interaction can start.

In a **service initiated interaction**, the service initiates the contact with the user. This can be done in an asynchronous way by just sending a message without any continuing interaction, or in a synchronous way with a continuing interaction, e.g. speech-based interaction. If there is an active Service Session already, which means that the user is already interacting with the service, this session and the according Access Session can be used to deliver the message or the interaction request. If there is no active Service Session, a possible suspended one can be resumed or a new one can be created. Then, it has to be checked, whether there is an active Access Session of the user, which then can be used for the interaction.

If there is no existing Access Session, a new one has to be created, which means that a connection to the user has to be established. According to available ambient information, e.g. user location, and user's personal preferences, a suitable terminal has to be chosen, e.g. the user's personal mobile phone. The Preferences Management System as well as the general system configuration should provide necessary information for such decisions.

3.2.4 General Functional Model of the Service Adaptation Framework

The purpose of the Service Adaptation Framework is to provide the necessary functions in order to realize I-centric User Interaction. These functions comprise the adaptation of the service's user interface to the particular characteristics of the Access Mechanism by considering user's preferences and available ambient information. Beside the adaptation of the service appearance, the service should adapt their behavior according to data provided by the Service Adaptation Framework, i.e. characteristics of the Access Mechanism, personal preferences, and ambient information.

Figure 3-6 shows a summary of the individual tasks of the Service Adaptation Framework depicted as use case.

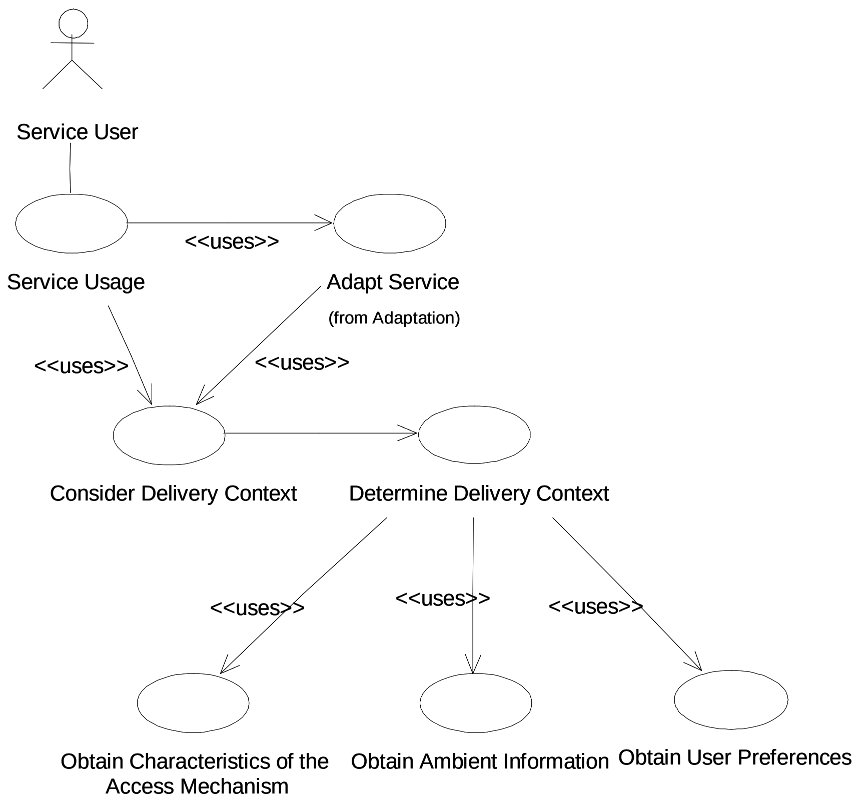


Figure 3-7: General Use Cases of the Service Adaptation Framework

The services, represented as *service usage*, and the Service Adaptation Function, represented as *adapt service*, consider the Delivery Context data, represented as *consider Delivery Context*. The Delivery Context is determined according to the identified user and the corresponding Access Session. It contains the characteristics of the Access Mechanism, related ambient information, and the preferences of the user, who is interacting with the service. Accordingly, the following main functional elements of the Service Adaptation Framework can be identified:

- Service adaptation function (GUIML interpreter and translators)
- Delivery context handler
- Capability manager
- Ambient information system
- Preference management system

Figure 3-8 depicts the relationships and the interworking of these functional elements.

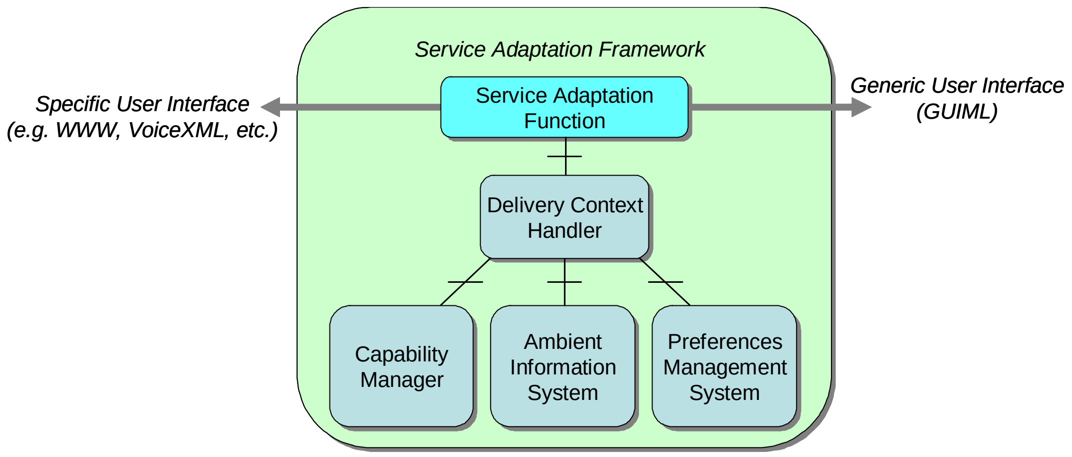

Service Adaptation Framework
Specific User Interface (e.g. WWW, VoiceXML, etc.)
Service Adaptation Function
Generic User Interface (GUIML)
Delivery Context Handler
Capability Manager
Ambient Information System
Preferences Management System

Figure 3-8: Main Functional Elements of the Service Adaptation Framework

The Service Adaptation Function translates the user interface, provided by the service, into the target user interface representation and vice versa. It depicts the core function of the I-centric User Interaction approach enabling the device independent, personalized, and ambient aware realization of the user-service interaction. In order to adapt the user interaction suitably, this function takes the current Delivery Context into account. The respective Delivery Context is compiled by the Delivery Context Handler, which in turn obtains the relevant data from the Capability Manager, the Ambient Information System, and the Preferences Management System.

The following sections introduce the functional elements in detail and explain their interworking, i.e. their interfaces. The following specifications are not dependent on any concrete implementation platform and depict, in this sense, a *Platform Independent Model* (PIM)[12]. According to the selected target platform (for development and execution), different *Platform Specific Models* (PSM) can be derived from it, which depict concrete implementation specifications. Chapter 4 contains an example of such a platform specific model, which was derived from this platform independent model, presented in this chapter.

3.2.5 Service Adaptation Function

The Service Adaptation Function transforms GUIML documents into the presentation of the target user interface technology and vice versa. The function takes the particular Delivery Context provided by the Delivery Context Handler into account. The complex transformation process is separated into individual tasks, which are represented as use cases in Figure 3-9.

[12] The concepts of Platform Independent Model and Platform Specific Model were introduced in OMG's Model Driven Architecture approach. See [OMGMDA].

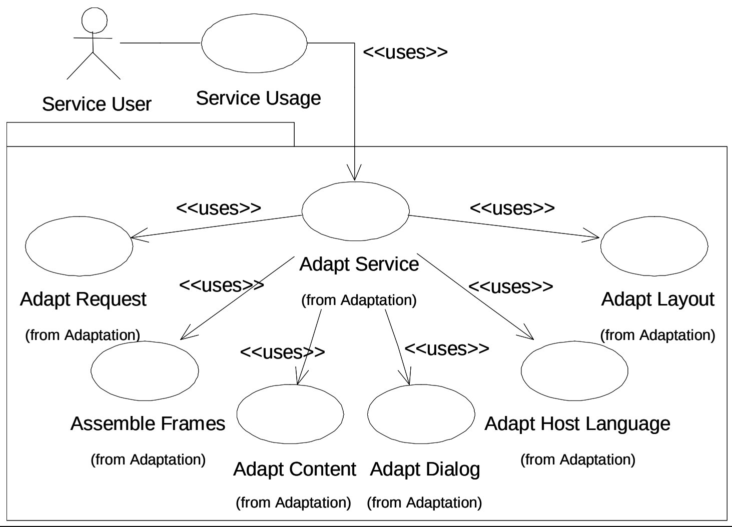


Figure 3-9: Service Adaptation Use Cases

The *service usage* describes the actual interaction between user and service. It contains the realization of the user interface. Because the service provides its user interface in a generic description, has to be transformed to the specific presentations, which is depicted by the *adapt service* use case. The transformation is a complex process, which utilizes further use cases for the individual transformation and adaptation tasks.

3.2.5.1 The Transformation Pipeline

Independent how the interaction was established and if the interaction is a dialog, i.e. synchronous, or just a message delivery, i.e. asynchronous, the transformation process of the Service Adaptation Function is always the same. The corresponding system elements have to ensure that the interaction requests are submitted to the Service Adaptation Function in the suitable way. The transformation process always starts with a request coming from the user. In service initiated scenarios, of course the initial request comes from the service. That is why, the system has first to establish the Access Session, i.e. connection to the user's terminal, then to issue the service's request from this Access Session. In this way, it is completely transparent for the Service Adaptation Framework, how the interaction was established.

The transformation process of the Service Adaptation Function is executed in a pipeline as illustrated in Figure 3-10.

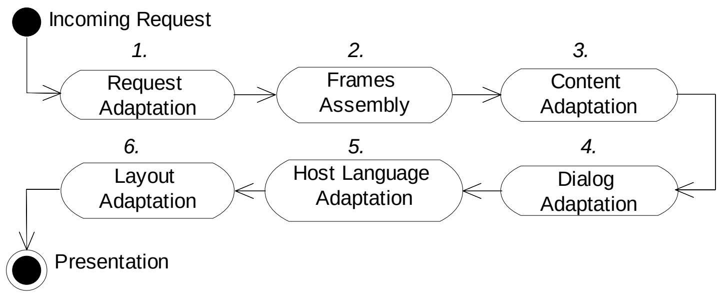


Figure 3-10: Transformation Pipeline Activity Diagram

1. First, the incoming request is analyzed. The request addresses a service (or rather the current context of the dialog) and can contain additional input parameters, e.g. data entered by the user. The parameters are adapted to the GUIML format and send to the service, which processes them and returns a GUIML document describing the next

dialog section. This contains the content to be presented (inline or be reference) and can contain additional references to external frames (common frames and service specific frames).

2. The GUIML document and the referenced frames are then assembled into a single document for further processing. The resulting document describes the complete content to be delivered to the user.
3. Now, the content has to be adapted to obtain a suitable presentation. This step consists of the selection of the right content pieces based on the specified Adaptation Switches and of the conversion of media elements, if necessary. This results in a subset of the GUIML document.
4. The documents still contains dialog descriptions specifying the input alternatives of the user. According to the target user interface system, these dialog fragments have to be adapted to the particular suitable presentation. This mostly relevant for non-hypertext-based presentation styles, such as speech-based systems, which need a special attention to provide suitable and usable dialogs to the user.
5. Then, the remaining document is translated to the suitable presentation according to the target user interface technology, i.e. to HTML, WML, or VoiceXML.
6. Finally, the layout of the content is adjusted according to the respective style sheets, if specified for the target user interface technology.

The individual activities are executed sequentially, one after the other. They modify the document, which is to be presented to the user, gradually. Whenever a transformer is active, the document is under its full control. Fundamentally, parallel processing is not applied, although real implementations should consider some optimizations in order to provide a sufficient runtime behavior.

To enable the interworking, the necessary input and the resulting output data have to be specified for each transformer function. Beside the actual document, each transformer function processes control data, which are obtained from the Delivery Context and from the transformer's own configuration database. In this way, document *D* is transformed to document *D'* by applying transformer *T* under consideration of control data *C*, i.e. D'=T(D, C).

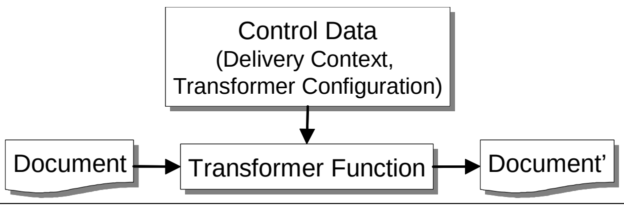


Figure 3-11: Transformer Function Overview

Figure 3-11 depicts the general overview how the transformer functions work. This template applies to all transformation functions of the Service Adaptation process.

3.2.5.2 Base Transformer Functions

According to the transformation pipeline (Figure 3-10), the required transformer functions can be identified as follows:

1. The *request adapter* processes the incoming requests, adapts included parameters, and calls the service.
2. The *frame assembler* includes external referenced frames in the document.
3. The *content adapter* adapts the content to a suitable presentation. It consists of:
   3.1. The *content selection*, which selects the most appropriate content from the alternatives specified in the GUIML document through the evaluation of the GUIML Adaptation Switches.

   3.2. The *content splitter*, which segments the content into smaller parts, if the content is too large to be presented on the target device at once. It encloses corresponding navigation elements to navigate through the individual segments.
   3.3. The *media converter* converts media to match the capabilities of the Access Mechanism.
4. The *dialog adapter* adapts the XForms based dialogs.
5. The *host language adapter* transforms the XHTML basic host language elements.
6. The *layout adapter* adapts the document to the specified layout and CSS presentation.

All transformer functions interpret relevant data, provided by the Delivery Context description, in order to provide a suitable adapted user interface. The following sections describe each base transformer function detailed and provide some exemplary data fragments, which are produced by the respective function. Furthermore, the overall sequence of the individual transformation steps is described in section 3.2.5.3.

Request Adapter

The request adapter forwards and adapts the requests coming from client applications to the services. The service content is loaded from the service provider's infrastructure. The deployment descriptor of each service assigns internal service identifiers to the real service address (URL). In this way, the request can be forwarded to the corresponding service instance.

The request adapter is needed, because there are differences in the format of the parameters that are sent by the terminal devices when the user inputs data. The format has to be adapted to be suitable to the service. For instance, when an HTML form is submitted, which contains checkboxes to select multiple items, the WWW browser sends separate name-value pairs for every selected value. A WAP browser behaves different in this situation. Instead, it sends a list of values separated by a semicolon in one name-value pair. A VoiceXML interpreter implements a checkbox in a complete different way. To enable the service to handle the submitted parameters regardless of the client the transformation of the parameters is necessary.

The request adapter transforms the submitted parameters to a standard format and forwards the request to the server hosting the service, which processes the input and returns the new content to be presented to the user. The returned content now can be processed further in the Service Adaptation pipeline in order to send the content in an appropriate presentation to the user.

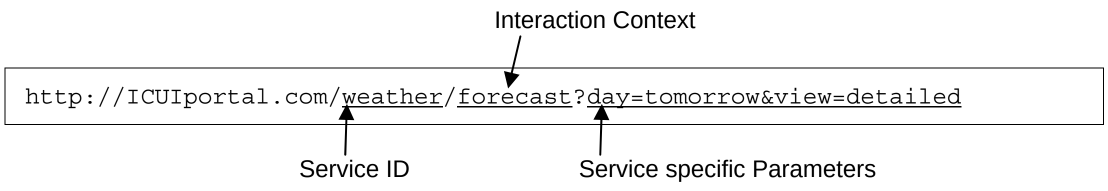
The request adapter receives a request to a service with service specific and Service Adaptation Function specific parameters. In the following examples, the common HTTP protocol is used, but the approach supports any suitable protocol.

```
http://ICUIportal.com/weather/forecast?ADAPTATIONSEG=3&day=tomorrow&view=
detailed
```

First, the Service Adaptation Function specific parameters are interpreted. The function searches the request for known parameters, e.g. the 'ADAPTATIONSEG' parameter. This parameter controls the navigation through the individual fragments, if the content has been segmented before by the content splitter. Accordingly, the next content segment is loaded from the local cache or from the service again. Otherwise, the request has to be forwarded to the service without the Service Adaptation specific parameters.

The Service Adaptation Function specific parameters have to be clearly marked in order not to confuse with the service specific parameters. A suitable naming schema should be chosen, by which these parameters could be sorted out clearly. The service specific parameters provided by the service should be inspected whether they conflict with adaptation function specific parameters and renamed (wrapped) accordingly, if necessary.

After the processing of the adaptation specific parameters, the request still contains the according service identifier, the current context of the interaction, and the service specific parameters. For example:



```
http://ICUIportal.com/weather/forecast?day=tomorrow&view=detailed
```

If the service utilizes a session model, the current context of the service interaction can also be described by an according session id, i.e.

```
http://ICUIportal.com/weather?SESSIONID=334747&day=tomorrow&view=detailed
```

From the service deployment descriptor, the real address of the service can be obtained and the request is send to the service as follows:

```
http://weather.com/forecast?day=tomorrow&view=detailed
```

The request adapter behaves as a normal client to the service supporting all necessary functions, such as cookies management, URL redirect, etc. The service processes this request according to its implementation and returns the next content to be presented to the user, i.e. a GUIML document, here depicted as a very simple example:

```
<document  xmlns:xfm=”http://www.w3.org/2002/01/xforms”
           xmlns:smil=”http://www.w3.org/2001/SMIL20/Language”
           xmlns:xlink=”http://www.w3.org/1999/xlink”>
   <head>
      <title>Weather Service</title>
   </head>
   <body>
      <frame href=”weather-menubar” />
      <frame id=”main” title=”Weather”>
         <h1>Weather Forecast</h1>
         <h2>Berlin, Sunday, 29 June 2003</h2>
         <smil:switch>
            <smil:ref type=”JPEG”
               src=”http://weather.com/forecast/images/weather-brd.jpg”/>
            <smil:ref type=”text” language=”en”
               src=”http://weather.com/forecast/weathertext-brd.txt” />      A
            <smil:ref type=”text” language=”de”
               src=”http://weather.com/forecast/weathertext-brd-de.txt” />
         </smil:switch>
         <xfm:model id=”weatherservice”>
            <xfm:submission action=”/OKSPortalWeatherService/Servlet”
                                 method=”get”/>
            <xfm:instance>
               <zipCode/>
            </xfm:instance>
         </xfm:model>
            <xfm:input model=”weatherservice” ref=”zipCode”>
               <xfm:label>
                   <smil:switch>
                          <smil:ref src=”data:Zip Code: “ language=”en”/>
                          <smil:ref src=”data:Postleitzahl: “
                                         language=”de”/>                      B
                   </smil:switch>
               </xfm:label>
            </xfm:input>
            <xfm:submit model=”weatherservice”>
               <xfm:label>
                   <smil:switch>
                          <smil:ref src=”data:Submit” language=”en”/>
                          <smil:ref src=”data:Senden” language=”de”/>
```

```
                    </smil:switch>
                </xfm:label>
            </xfm:submit>
      </frame>
   </body>
</document>
```

The example above contains a user interface description consisting of a frame reference and an inline frame (‘main’), which describes the content, i.e. the weather forecast, to be presented to the user (section ‘A’ in the example) and a possibility to input a zip code of another location (section ‘B’ in the example). The content provides a textual (‘weathertext-brd.txt’) and a graphical (‘weather-brd.jpg’) alternative.

Possible service specific parameters contained in the document have to be inspected and, if necessary, to be encoded to avoid ambiguities. Then, this raw GUIML document is provided as the output of the request adapter function.

Frame Assembler

The document obtained from the request adapter is described in GUIML and can therefore contain references to external frames beside the inline content, as shown in the example above. The frame assembler has to resolve the external references and to include the complete content into the document, i.e. the document does not contain any reference to frames thereafter.

As described in section 3.5.4.4, frames have to be transformed in different ways depending on the user interaction technology and terminal type. For instance, frames can be used in HTML without any problem, but in speech-based interfaces, i.e. VoiceXML, or WAP-based interfaces, they have transformed into other suitable presentations. Therefore, the fame assembler function has to utilize according templates. Which template has to be chosen is derived from the Delivery Context. Each template specifies conditions, which have to be evaluated first. Then, according to the selected template, the frames are resolved. Table 3-1 describes some examples of templates and their conditions.

Conditions	***Template***
mime-type=text/vxml	voicexml.template
mime-type=text/vnd.wap.wml	wml.template
mime-type=text/html screenwidth<200 screenheight<150	htmlPDA.template
mime-type=text/html	html.template

Table 3-1: Exemplary Frame Substitution Templates

In the example, the referenced frame (‘weather-menubar’) is provided by the service and is therefore denoted as a service-frame. According to the system policies, additional frames denoted as portal frames can be included into the document as follows:

```
...
   <body>
      <frame id=”menu” href=”ICUIportal.com/home”/>
      <frame href=”ICUIportal.com/myFavourites”/>
      <frame href=”weather-menubar”/>
      <frame id=”main” title=”Weather”>
   ...
   </body>
...
```

This is necessary, in order to provide not only the service’s content, but also additional navigation, such as service selection, etc., to the user. In addition, the personal settings according the structure of the content can be included in this way, e.g. services shortcuts or a combined presentation.

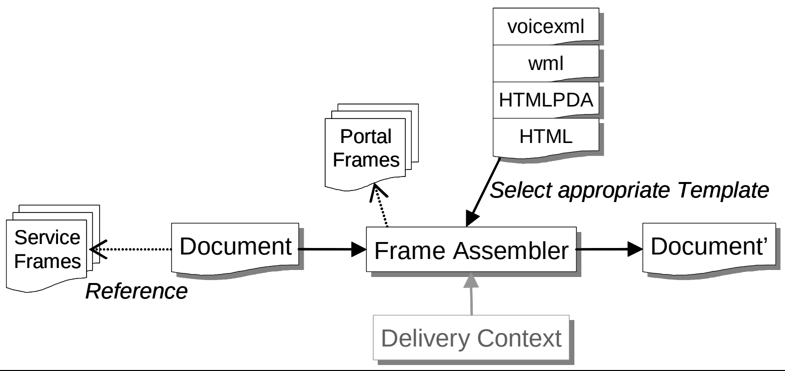


Figure 3-12: Resolution of referenced Frames

The frame assembler, as shown in Figure 3-12, has to fetch the external contents of the references found in the document, to select the correct template, to transform the content according to the template, and to provide them altogether in a single document as shown in the following example[13]:

```
<document  xmlns:xfm="http://www.w3.org/2002/01/xforms"
           xmlns:smil="http://www.w3.org/2001/SMIL20/Language"
           xmlns:xlink="http://www.w3.org/1999/xlink">
   <head>
      <title>Weather Service</title>
   </head>
   <body>
      <frame id="weather-menubar" title="Weather Menu"
              style="background-image:url(weatherbar.gif)">
         <xfm:trigger model="menubar">
            <xfm:label>
               <smil:switch>
                   <smil:ref src="homeweather.gif"/>
                   <smil:ref src="data:Weather at home" language="en"/>
                   <smil:ref src="data:Wetter daheim" language="de"/>
               </smil:switch>
            </xfm:label>
            <xfm:action>
               <xfm:load xlink:href="home"/>
            </xfm:action>
         </xfm:trigger>
         <xfm:trigger model="menubar">
            <xfm:label>
               <smil:switch>
                   <smil:ref src="worldweather.gif"/>
                   <smil:ref src="data:World Weather" language="en"/>
                   <smil:ref src="data:Weltwetter" language="de"/>
               </smil:switch>
            </xfm:label>
            <xfm:action>
               <xfm:load xlink:href="world"/>
            </xfm:action>
         </xfm:trigger>
         <xfm:trigger model="menubar">
            <xfm:label>
               <smil:switch>
                   <smil:ref src="details.gif"/>
                   <smil:ref src="data:Detailed View" language="en"/>
                   <smil:ref src="data:Detailansicht" language="de"/>
               </smil:switch>
            </xfm:label>
            <xfm:action>
```

[13] The elements indicated with a “+“ in the first row are hidden for better readability purposes.

```
               <xfm:load xlink:href="details"/>
            </xfm:action>
         </xfm:trigger>
      </frame>
      <frame id="main" title="Weather">
         <h1>Weather Forecast</h1>
         <h2>Berlin, Sunday, 29 June 2003</h2>
+        <smil:switch>
+        <xfm:model>
+        <xfm:input>
+        <xfm:submit>
      </frame>
   </body>
</document>
```

In the example above, the references to the service frame ('weather-menubar') was resolved and the content of this frame is now contained inline. Apart from the media elements, the resulting document now contains the complete content and can be processed as a whole by the following processing steps.

Content Adapter

The content adapter prepares the content, described by the GUIML document, for the presentation to the user. The appropriate media elements have to be selected from the alternatives as specified in the GUIML document. Furthermore, these media elements have to be adapted to fit to the target user interface technology and to the capabilities of the Access Mechanism, i.e. concerning screen resolution and network transfer rate, etc. Therefore, the content adapter function consists of the following three sub-functions:

- Content selection
- Content slitter
- Media converter

These sub-functions are introduced in the following sections.

Content Selection

This component searches all Adaptation Switches in the document, i.e. certain attributes and SMIL switch statements. It evaluates the conditions against the Delivery Context data. For those, which are evaluated successfully, the corresponding content remains part of the document. The other alternatives are removed. In this way, all Adaptation Switches are resolved and the content, which should not be presented to the user, is removed from the document.

In the example, there are a number of content alternatives, such as:

```
<smil:switch>
   <smil:ref type="JPEG"
         src="http://weather.com/forecast/images/weather-brd.jpg"/>
   <smil:ref type="text" language="en"
         src="http://weather.com/forecast/weathertext-brd.txt"/>
   <smil:ref type="text" language="de"
         src="http://weather.com/forecast/weathertext-brd-de.txt"/>
</smil:switch>
```

First, the attributes of all elements are evaluated. Two of the three alternatives in the example have a language attribute. According to the preferred language of the user and to the support of the terminal, the appropriate alternative has to be selected. The necessary data, i.e. preferred and supported language, should be described in the Delivery Context. Assuming that the preferred and supported language is English, the example is transformed to:

```
<smil:switch>
   <smil:ref type="JPEG"
```

```
        src=”http://weather.com/forecast/images/weather-brd.jpg”/>
   <smil:ref type=”text” language=”en”
        src=”http://weather.com/forecast/weathertext-brd.txt”/>
</smil:switch>
```

In the next step, the remaining alternatives have to be resolved. In the example, the service provides still two alternatives — a JPEG picture and a textual representation. If the target user interface cannot present JPEG pictures, the textual representation is selected and the reference to the JPEG picture is removed. If all options are evaluated and the most suitable alternative is determined, the ‘switch’ statement is removed, likewise.

```
<smil:ref type=”text” language=”en”
      src=”http://weather.com/forecast/weathertext-brd.txt”/>
```

If the terminal is able to display images but not encoded as JPEG, e.g. WAP devices, also the text would be selected. Nevertheless, the service can enforce the suitable adaptation of the content by adding a corresponding adaptation switch as follows. This would result in the selection of the graphical representation, i.e. by conversion, instead of the textual alternative:

```
<smil:switch>
   <smil:ref type=”JPEG” systemCapable=”image”
        src=”http://weather.com/forecast/images/weather-brd.jpg”/>
   <smil:ref type=”text” language=”en”
        src=”http://weather.com/forecast/weathertext-brd.txt”/>
</smil:switch>
```

Content Splitter

Terminal devices have different presentation capabilities, such as the screen dimension, and there are additional restrictions of the different user interface technologies, such as maximal size of the user interface document. That is why the content has to be examined accordingly. The content splitter evaluates, if the complete content can be presented at once in a practical and user-friendly way. For example, a WAP terminal has usually a small display and the size of a WAP deck is restricted (1400 byte binary data), likewise. Therefore, the content, which can be presented on a desktop computer in a WWW browser without any problem, cannot be presented on a WAP terminal without any modifications. This is also relevant with regard to usability aspects. For instance, speech-based interfaces should not play back extensive content, because the user cannot navigate through it easily in contrast to graphical interfaces.

If necessary, the content has to be divided into segments, which are then presented sequentially. According to the restrictions of the target user interface technology and according to the constraints derived from the user-friendliness demand, the most appropriate size of the segments has to be selected. If the document is split up, corresponding navigation components have to be inserted. The navigation and the adaptation are controlled through special request parameters, which are to be treated by the request adapter in different way than the service’s own parameters. The content splitter should consider the XForms[14] group element, by which the service author can describe parts of the user interface, which shall not be divided.

The following example contains text, which is too long to be presented at once:

```
<smil:ref src=”data:A very long text.”/>
```

[14] As described in section 3.5.4, GUIML uses XForms technology [W3CXFORMS] to describe the dialogs in a generic, device-independent way.

Therefore, the text element has to be segmented into several pieces. This means, only the first segment is included and an according navigation element is additionally included, by which the user can request the next segment:

```
<smil:ref src="data:A very ..."/>
<xfm:trigger model="next">
   <xfm:label>
      <smil:switch>
         <smil:ref src="next.gif"/>
         <smil:ref src="data:Next >>" language="en"/>
         <smil:ref src="data:Weiter >>" language="de"/>
      </smil:switch>
   </xfm:label>
   <xfm:action>
      <xfm:load xlink:href="next?ADAPTATIONSEG=2"/>
   </xfm:action>
</xfm:trigger>
```

By providing suitable navigation segments, i.e. start, previous, and next, the user can comfortably browse through the complete content element even on devices, which are not able to present the complete content at once. In addition, this procedure is very useful for speech-based applications.

Beside the separation of content into smaller pieces, the content can be shortened by innovative 'summarizing' algorithms, likewise. Various research projects have developed already first approaches to analyze text and audio content to characterize the importance of individual segments, i.e. by suitable rating mechanisms. According to the required target size, a new media element can be compiled by copying the appropriate number of segments. In order to obtain the semantic as good as possible the segments with the highest importance rating are selected first.

Media Converter

Because of the different display and processing capabilities, media elements may have to be adapted accordingly. The adaptation can be realized by suitable conversions. The media converter function enables the conversion of media elements by evaluating their properties. It provides the media format, e.g. GIF to JPEG, and the media type conversion, e.g. TIFF to text. The following conversions are needed:

- *Picture to Picture*: conversion of size, color depth, and format
- *Video to Video:* conversion of bit rate, size, color depth, and format
- *Video to Picture:* extraction of still pictures
- *Video to Audio:* extraction of audio tracks
- *Audio to Audio:* conversion of sample rate and format
- *Audio to Text*: speech recognition
- *Text to Audio*: speech synthesis

Partially, some conversion support may be provided by other system elements, likewise. For example, in speech-based system the corresponding gateway usually implements the complete speech recognition and speech synthesis support. For such systems, the Service Adaptation Function should provide the content suitable to these gateways, i.e. not converting itself but providing the raw data format to be converted by the gateways. However, this depends on the actual implementation and on an appropriate configuration of the media converter function.

The media converter function analyses the properties of all media elements remained in the document. Examples for properties are the type of the media element, its size, or its number of colors. The function considers the capabilities of the Access Mechanism as well as the user's preferences to take the decision. If the media element fits, a corresponding reference remains in the document as it is.

If the media element does not fit directly, conversion possibilities have to be inspected. The achievable properties of all media elements are determined and evaluated again to obtain the necessary conversion strategy. The achievable properties are looked up in a registry of available converters. Accordingly, a reference with the necessary conversion parameters and the source reference are included into the document. If there is no suitable conversion strategy found, a plain text representation, i.e. taken from the ‘alt’, ‘longdesc’, ‘abstract’, or ‘title’ attributes if available, is inserted. In the following example, the image shall be converted to a wireless bitmap in order to be presented on a WAP terminal:

```
<smil:ref type=”wbmp”
   src=”ICUIportal.com/mediaconverter?mimetype=image/vnd.wap.wbmp&width=5
0&src=http://weather.com/forecast/images/weather-brd.jpg”/>
```

When this reference is request, the media converter is activated automatically. It loads to original media element from the given source and performs the conversion according to the conversion parameters, i.e. requested mime-type and media characteristics. The converted content is returned instead of the original media element.

Because media conversion is a time and resources consuming process, suitable optimization concepts, such as caching algorithm, should be applied in real systems. In this way, the media converter can be triggered as soon as the conversion link is included into the document even before the complete adaptation process is finished and send back to the user’s terminal. In this case, the converter should temporarily store the converted content until the media elements are actually requested from the terminal.

Dialog Adaptation

Beside the content adaptation, the adaptation of the dialog, i.e. the navigation through the content, is one of the most critical tasks in the device independent user interaction. The different user interface technologies have dissimilar support for navigation. Whereas in WWW, any number of forms and links can be used concurrently, in WAP and speech-based user interfaces, the number is limited for technical reasons, but above all for usability reasons. In HTML pages, the user can easily survey all provided feedback possibilities and in this way can choose easily among them. The interaction schemas in other user interface technologies, such as WAP and speech, are rather sequentially oriented because of the limited presentation and interaction capabilities.

According to the support of the dialog presentation, the different user interface technologies therefore have to be differentiated into:

- User interface technologies with *sequential dialog* presentation, which can present only a few feedback possibilities, such as WAP and speech-based user interfaces
- User interface technologies with *parallel dialog* presentation, which can present a larger extend of feedback possibilities, such as WWW pages

If the dialogs, which are described in a GUIML document, would be translated into the target user interface presentation without any special consideration of those differences, the resulting user interface would be unusable or at least uncomfortable to the user. The arrangement of the content into frames has to be taken into account to solve this issue. If necessary, only the dialog of the ‘main’ frame should be presented directly. The content of other frames should be presented indirectly, e.g. hidden in a corresponding menu structure. Especially, speech-based user interaction has to be designed carefully in order to provide a natural as possible presentation, because of its sequential nature (only few response options can be provided to the user at a time).

Therefore, for each target user interface technology and for the individual terminal types, according rule sets have to be defined:

- For *sequential dialog presentation*: The focus remains on the content of the ‘main’ frame, which should contain the most relevant content. Therefore, the dialog of the ‘main’ frame is transformed whereas the dialogs of additional frames are hidden in an extra menu option. This means, the frames are separated and this extra menu option provides the navigation to the other frames. This means, the respective content and dialog of these frames are not included, but only references to them. In this way, the user can reach the dialogs of the additional frames by opening this extra menu. The service author has to ensure a reasonable adaptation of the dialogs by using a logical frame structure. However, as a principal guideline, the dialogs should be kept simple as possible to allow meaningful presentations in different user interface technologies.
- For *parallel dialog presentation*: According to the capabilities of the target user interface technology, all frames including their dialogs are transformed coherently. This means, the dialogs of the individual frames are presented at once. There is no need for special sorting or ordering.

This means, the general procedure is that the ‘main’ frame remains as it is, i.e. it is translated without additional modifications. For limited user interface technologies, such as WAP and speech-based interaction, additional frames are removed from the document and according access (references) to them are included instead, preferably in menu structures. In this way, these references can be translated for example into the ‘option menu’ in WAP or as ‘link elements’ in VoiceXML. However, this means that additional navigation elements are added to the actual user interface provided by the service. These additional elements have to be embedded smoothly, so that the user does not recognize the restructering of the navigation.

For example, when adapting the dialog for VoiceXML-based speech interaction systems, additional prompts have to be inserted to guide the user through the possible navigation. It is important to choose the right language for these prompts depending on the user’s preferences, on the language used by the service, and on the support of the speech interaction system.

Host Language Adapter

The host language adapter function converts the remaining GUIML document into the concrete user interface description language. This means, after this step, no GUIML elements reside in the resulting document anymore. The document is then fully complaint with the target user interface technology.

Therefore, for each user interface technology supported, according transformation rules have to be defined. This is basically a straightforward process, because all the critical adaptation decisions are already made in the previous transformation steps. The most difficult task is the removal of nested tags that are allowed in GUIML but not in some user interface markup languages such as WML and VoiceXML. The transformation of the general XHTML Basic and SMIL elements is quite simple, whereas for the mapping of dialog elements various transformation alternatives are possible partially.

The XForms controls are adapted to the corresponding user interface controls of the devices in a one-to-one mapping. If necessary, an XForms control may possibly be adapted to a functional equivalent. The conversion can be difficult in certain cases, because XForms allow multiple usages of its controls by assigning multiple actions to them, which is not supported by all user interaction technologies and according devices. However, the host language adapter function supports sufficient controls to build functional user interfaces. The GUIML elements *trigger* and *select1* should support multiple functions depending on the XForms actions *send* and *load*, i.e. they could submit a form, link to a new page, or in case of select1 let the user select one of multiple possibilities. Table 3-2 gives an overview of the general tag conversion showing the possible XForms controls conversion to HTML, WML, and VoiceXML.

XForms	*HTML*	*WML*	*VoiceXML*
input	input type=”text”	input	field
secret	input type=”password”	input	field
textarea	textarea	input	field
trigger	href input type=”submit”	href, go	next submit
submit	input type=”submit”	go	submit
select1	href, input type=”submit” input type=”radio”	go go option	next submit field
select	input type=”checkbox”	option multiple=”true”	field

Table 3-2: Adaptation of XForms Controls

Layout Adaptation

The layout adaptation is the last transformation step. It takes the device’s presentation capabilities and user’s preferences into account. The devices, which are able of displaying multiple frames at once, i.e. WWW terminals, can be provided with device and user specific layout configuration, e.g. using Cascading Style Sheets (CSS). The layout adapter uses this configuration to assign the frames to the user interface layout. The layout support is a necessary function in order to realize handsome interfaces. However, only HTML currently supports this, but not each browser program. WAP and VoiceXML do not provide a support for this at all.

The layout can be specified by the user as a personal preference, e.g. in dependence on the terminal, location, etc. The system can provide suitable tools to specify the layout or can provide a selection of predefined layout sets, from which the user can choose. In this way, the appearance of the user interface can be personalized.

3.2.5.3 Adaptation Sequence Diagram

The following UML Sequence Diagram describes more detailed the adaptation process executed by the Service Adaptation Function.

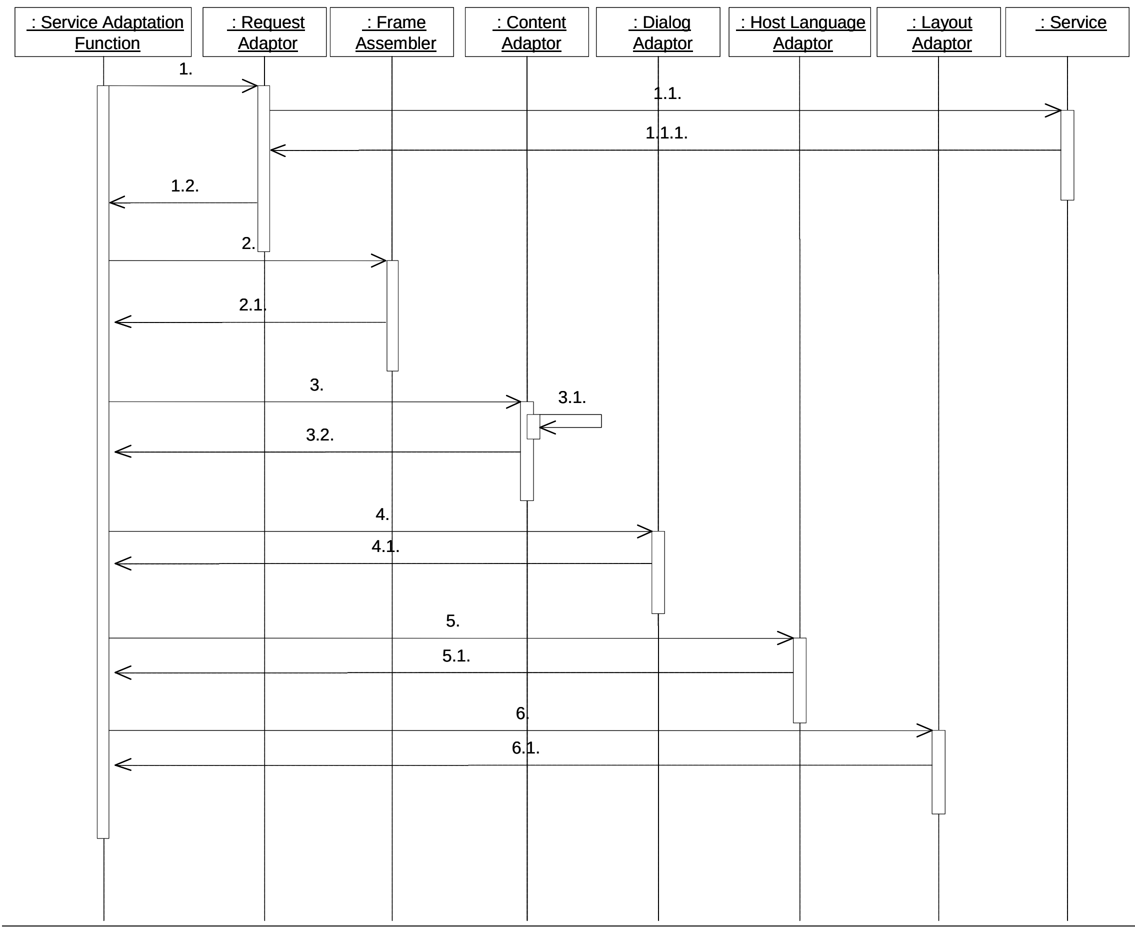


Figure 3-13: Sequence Diagram of the Adaptation Process

When a terminal device sends a request to the Service Adaptation Function, a suitable transformation pipeline is selected according to the valid Delivery Context. Then, the transformation pipeline is started. After adapting the request parameters (1), the service page is loaded from the service (1.1). The returned GUIML document (1.1.1) is the base for further transformation steps. The frame assembler is called at next (2). After the frames are included into the GUIML document (2.1), the content adaptation is executed (3). The content adaptation is executed in three steps (3.1) as depicted in Figure 3-14.

Now, three transformation steps follow: adapting the dialogs (4), the host language (5), and the layout (6). The document is now available in the specific user interface format and is sent back to the accessing terminal device.

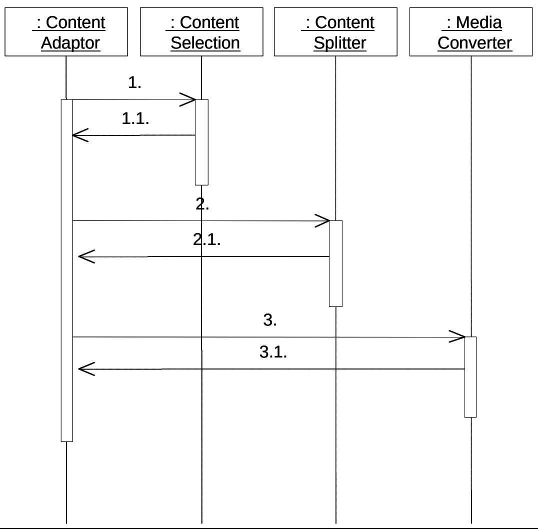


Figure 3-14: Sequence Diagram of the Content Adaptation

The content adapter component uses three sub functions. First, it calls the content selection component (1), which evaluates the Adaptation Switches and selects the most suitable media element. Then, the content splitter is called (2), which determines, if the content has to be segmented for the presentation. At last, the media converter is called (3), which evaluated whether the media elements have to be converted for the presentation and which includes according conversion request into the document.

3.2.6 Delivery Context Handler

The Delivery Context describes all data to control the Service Adaptation process and the service execution. It includes the capabilities of the Access Mechanism, the user's preferences, and available ambient information. The Delivery Context Handler manages the Delivery Context data and provides them to the Service Adaptation Function and to the services. The context data are obtained from the corresponding components: the Capability Manager, the Ambient Information System, and the Preferences Management System.

The Capability Manager gathers all data, which describe the capabilities of the Access Mechanism. It should support a number of relevant protocols and concepts for capabilities negotiation, such as CC/PP for example. Beside the capabilities and properties, the Capability Manager also records data from the network, i.e. bandwidth, transfer rate, etc. It provides the data using the XDCL language as introduced in section 3.5.6.

The Ambient Information System gathers and manages ambient information. Source of ambient information can be complete sensor networks, sensors in the terminal device, or any kind of technology able to sense physical data. The Ambient Information System itself interprets the raw data and provides revaluated information. Generally, all ambient information is assigned to a location context, i.e. the ambience, or alternatively to a user. By providing an according location description, i.e. position and extend, or a user identifier, the Ambient Information System is able to provide all related ambient information. This means, the Delivery Context Handler can ask the Ambient Information System for the ambient information, which is related to a current communication session by specifying the location information or the user of the communication session. The location information as well as the user identification, if available, should be stored in the Access Session.

Furthermore, the Delivery Context Handler interrogates the Preferences Management System about the user's preferences. Using the interfaces as provided by the Preferences Management System, the Delivery Context Handler can quite simple obtain all the user's preferences, which are related to the current situation. The Preferences Management System itself needs also access to other data, which are to be described by the Delivery Context in order to evaluate the Selection Context of the different profiles.

A Delivery Context is to be created whenever a communication session starts, i.e. whenever there is an access or a Service Session. As Access Sessions as well as Service Sessions are always linked to a user, the Delivery Context is always linked to a user. This means, the Delivery Context Handler creates a Delivery Context on request, assigns an identification, which is further on referenced in the Access and Service Sessions. In this way, the components, which need to access the Delivery Context data can obtain the identification from the sessions and interrogate the Delivery Context Handler accordingly. Of course, the Delivery Context is not static. Instead, the data, especially the ambient information, changes over time. This means, the identification of a created Delivery Context refers rather to a virtual data structure, which is kept up-to-date and whose data can change continuously.

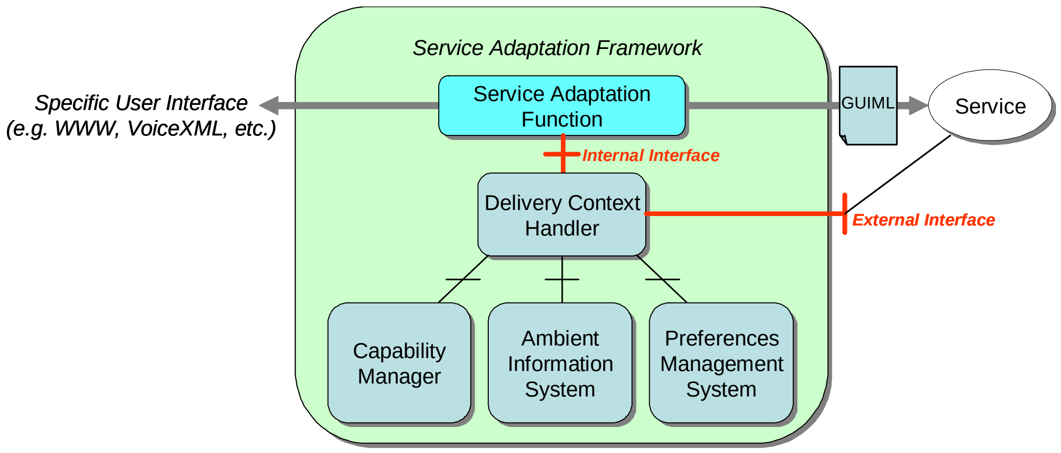

Service Adaptation Framework
Specific User Interface
(e.g. WWW, VoiceXML, etc.)
Service Adaptation Function
GUIML
Service
Internal Interface
Delivery Context Handler
External Interface
Capability Manager
Ambient Information System
Preferences Management System

Figure 3-15: The Interfaces of the Delivery Context Handler

As shown in Figure 3-15, the Delivery Context Handler provides two interfaces to access the Delivery Context data. There is one interface, which is denoted as the *internal* interface and primarily provided to the Service Adaptation Function. Beside the Service Adaptation Function, the services themselves have an interest in Delivery Context data in order to appear personalized and ambient aware. Because of the different roles, an additional interface is provided to the services. This interface, denoted as the *external* interface, contains additional functions in order to store data to the Preferences Management System. It should be implemented using suitable distributed software technologies, e.g. SOAP, CORBA, or RMI, in order to enable a remote access, because the services will run on different hosts. According to the implementation platform of the I-centric User Interaction system, the internal interface does not necessarily have to be accessible from remote, i.e. it could be implemented as an API supporting local functions calls only.

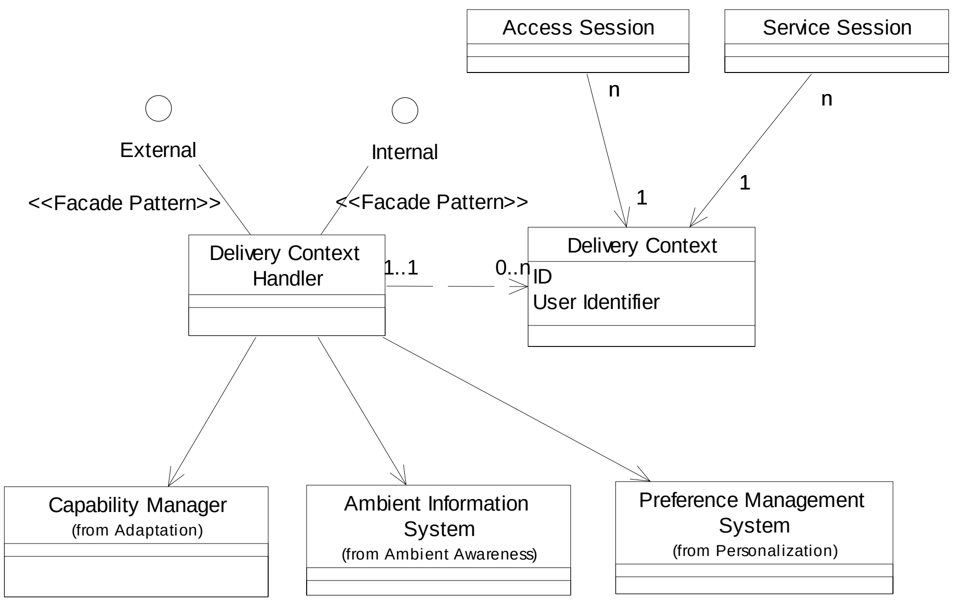


Figure 3-16: Delivery Context Management

Figure 3-16 shows an overview how the Delivery Context Handler manages and provides access to a Delivery Context. The Delivery Context Handler creates and manages specific Delivery Contexts by retrieving the necessary data from the corresponding components. A Delivery Context contains therefore all data, which are related to the current communication session, i.e. according to the access or Service Sessions. Dependent on what data are available, each Delivery Context therefore consists of:

- The capabilities of the Access Mechanism as provided by the Capability Manager
- The ambient information provided by the Ambient Information System
- The user’s preferences provided by the Preferences Management System

These data have to be provided through the identified interfaces, i.e. the internal as well the external interface. The usage, i.e. the retrieving, of the Delivery Context is the same for both interfaces, only the change of the data has to be restricted in the external interface. This means that the Delivery Context Handler has to take care that only the relevant data, e.g. the correct user preferences, are included in a Delivery Context.

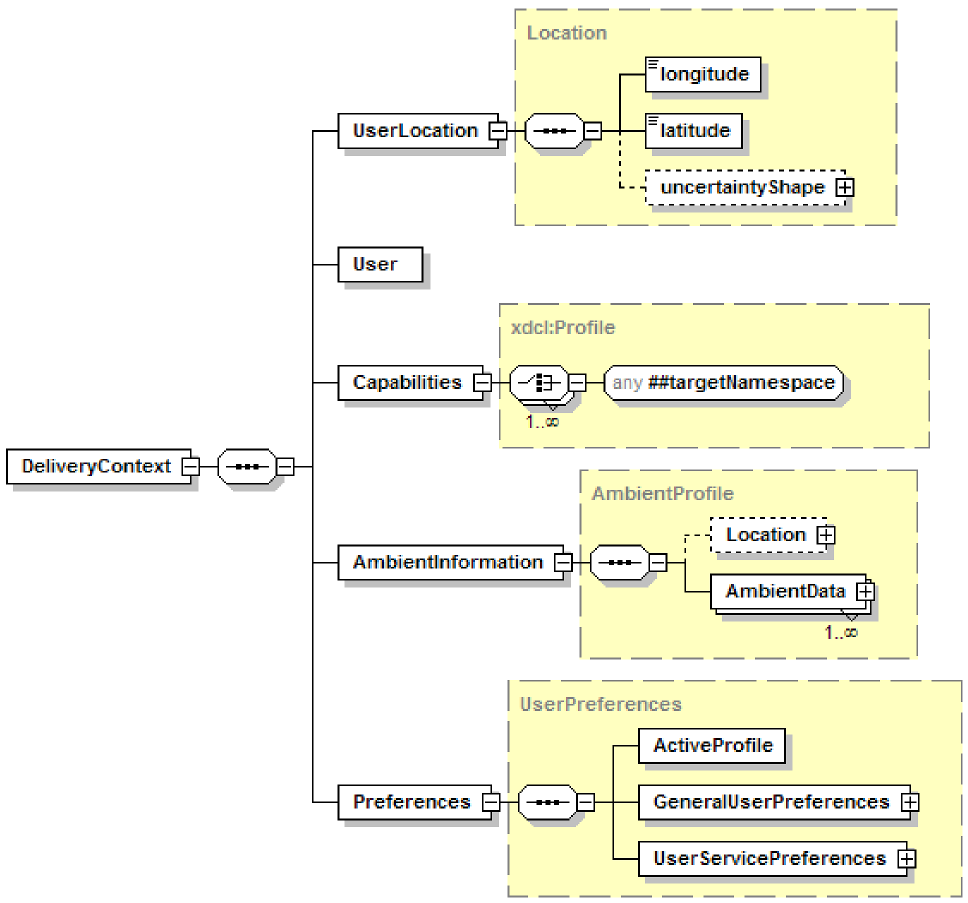


Figure 3-17: XML Schema Specification of the Delivery Context

Figure 3-17 shows the structure of the Delivery Context described by XML schema specification[15]. According to this specification, the complete data, which depict the whole Delivery Context, can be presented in a single XML document. Each Delivery Context is characterized by a location, i.e. user location element, a user identifier, i.e. the user element, and the actual Delivery Context data, i.e. capabilities, ambient information, and preferences elements. Such an XML document is to be created by the Delivery Context Handler and to be provided through its interfaces by according 'get' methods. Therefore, the Delivery Context Handler provides a method to receive the current Delivery Context described in the proposed XML structure.

As defined by the facade design pattern[16], the Delivery Context Handler depicts a facade object, which unites and combines the functionalities of the Capability Manager, the Ambient Information System, as well as the Preferences Management System. As described above, the Delivery Context Handler is requested to create a virtual Delivery Context when an Access or Service Session is being created. This means, the Delivery Context Handler has to maintain these virtual Delivery Contexts as long as the sessions exist. Additionally, this means that for Delivery Context query that the according identification of the Delivery Context has to be provided to the Delivery Context Handler. Therefore, the method to query the current Delivery Context can be defined as follows:

```
getDeliveryContext(DeliveryContextIdentifier):DeliveryContextDoc
```

[15] The complete Delivery Context specification can be found in the appendix, section 7.2.

[16] See [GAMMA] for detailed description of the applied design patterns.

This method returns an XML document describing the requested Delivery Context addressed by Delivery Context identifier. The returned document contains all available Delivery Context data coherently. Accordingly, it can be evaluated by the calling client.

The clients, which query the current Delivery Context, can also provide data, which is to be included in the Delivery Context. For example as described in the Personalization model (section 3.3), the service gathers the user's preferences and has to provide these to the Preferences Management System. Therefore, in addition to the query of the Delivery Context, the Delivery Context Handler has to provide according support to store data to the Delivery Context. However, this is mainly relevant for the personalization data.

The Capability Manager determines the capabilities of the Access Mechanism directly, which means that the different gateways derive the necessary data and provide them to the Capability Manager directly. There is no need to provide these data through the Delivery Context Handler, which therefore concentrates only on the interrogation of the Capability Manager and the integration of the obtained data into the Delivery Context document.

This also applies to the gathering of ambient information, which can be provided by sensor networks, but can be also derived from the Access Mechanisms, i.e. from the terminal directly or from the network. Again, the responsible gateways resolve these data and supply them to the Ambient Information System directly.

This means that the Delivery Context Handler has to provide the whole Delivery Context and the access to the Preferences Management System. Accordingly, the interfaces consists of the 'getDeliveryContext()' method and the methods inherit from the according interfaces of the Preferences Management System. Because the Delivery Context Handler needs the Delivery Context identification in each method call, methods from the preference management interfaces cannot be inherited directly, but are defined identical with an additional parameter for the Delivery Context identifier.

In this way, the Delivery Context Handler realizes a facade to the Preferences Management System. According to the Delivery Context addressed, i.e. the given identifier, these method calls are forwarded to the Preferences Management System. The external interface redefines the methods from the external interface of the Preferences Management System. The internal interface is mainly to be used by the Service Adaptation Function and does not need to provide methods to store data. It therefore only contains a method to obtain the Delivery Context.

3.2.6.1 Internal Interface

As shown in Figure 3-18, the internal interface of the Delivery Context Handler provides only one method to retrieve the complete Delivery Context. The returned document corresponds to the structure as described in Figure 3-17.

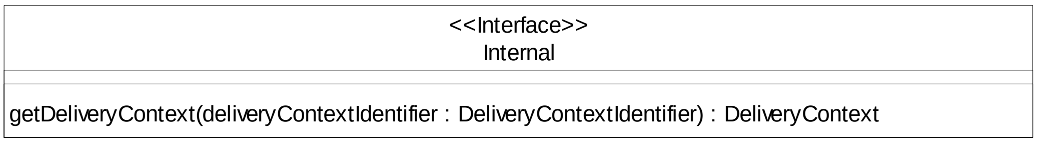

<<Interface>>
Internal
getDeliveryContext(deliveryContextIdentifier : DeliveryContextIdentifier) : DeliveryContext

Figure 3-18: The internal Interface of the Delivery Context Handler

The 'getDeliveryContext()' method returns the current Delivery Context according to the given identifier. It is up to the implementation, if the other components, i.e. the Capability Manager, the Ambient Information System, and the Preferences Management System, are called during the execution of this method or if the Delivery Context Handler maintains permanently an up-to-date cached copy, which is returned instead. Whereas the first approach might delay the execution and depict therefore a possible bottleneck influencing the performance, the second approach enables an immediate processing and a quick return of the data.

3.2.6.2 External Interface

The external interface of the Delivery Context Handler is to be provided to the services. Using this interface, services can request the current Delivery Context and can also store preferences data. As shown in Figure 3-19, the external interface inherits from the internal interfaces and provides additional required methods.

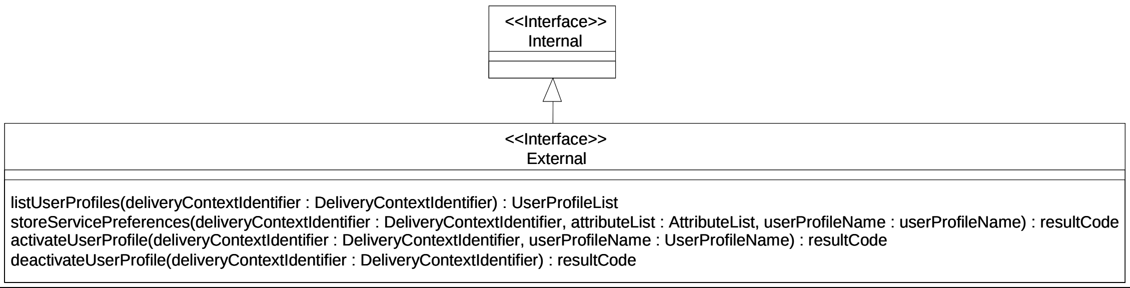

<<Interface>>
Internal
<<Interface>>
External
listUserProfiles(deliveryContextIdentifier : DeliveryContextIdentifier) : UserProfileList
storeServicePreferences(deliveryContextIdentifier : DeliveryContextIdentifier, attributeList : AttributeList, userProfileName : userProfileName) : resultCode
activateUserProfile(deliveryContextIdentifier : DeliveryContextIdentifier, userProfileName : UserProfileName) : resultCode
deactivateUserProfile(deliveryContextIdentifier : DeliveryContextIdentifier) : resultCode

Figure 3-19: The external Interface of the Context Delivery Handler

Services use the ‘getDeliveryContext()’ method to obtain the complete context delivery data including the relevant user preferences. If a service needs to change the preference data, because the user has specified a different setting during the service usage, the service uses the additional methods, which are derived from the Personalization model, e.g. the ‘storeServicePreferences()’ method. These additional methods are described detailed in section 3.3.5.

3.2.7 Capability Manager

The Capability Manager determines the capabilities and characteristics of the Access Mechanism. Because, an I-centric User Interaction system supports different kinds of access networks and communication services, the methods to derive the according capabilities are different. This means, the gateways depicting the interface of the I-centric User Interaction system to the network and to the user’s terminal, have to capture the capabilities in the technology specific way and to provide these data to the Capability Manager, which translates them into a common format. The Capability Manager has therefore to support a number of protocols and concepts to support the different Access Mechanisms.

The capability data are gathered depending on the actual system. For example, there might be a CC/PP[17] capable gateway, which receives CC/PP profiles from the terminals. If the terminals do not support CC/PP or similar protocols, the gateways should implement different methods to gather the capabilities, e.g. HTTP content negotiation.

[17] CC/PP: Composite Capabilities Preferences Profile [W3CCCPP]. Please refer to the introduction of the models for Personalization, for Ambient-Awareness, and for Generic User Interaction for further details.

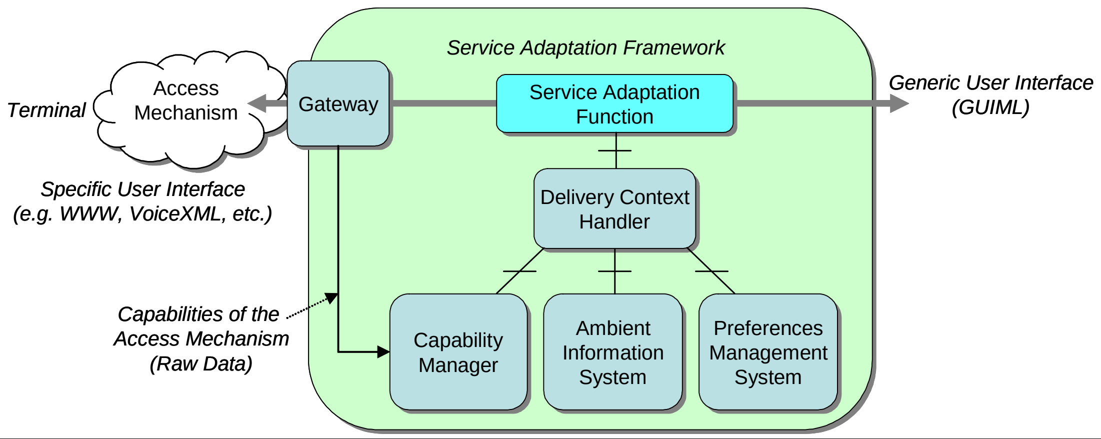

Service Adaptation Framework
Terminal
Access Mechanism
Gateway
Service Adaptation Function
Generic User Interface (GUIML)
Specific User Interface (e.g. WWW, VoiceXML, etc.)
Delivery Context Handler
Capabilities of the Access Mechanism (Raw Data)
Capability Manager
Ambient Information System
Preferences Management System

Figure 3-20: Gateways capturing Capabilities of the Access Mechanism

As shown in Figure 3-20, the gateways capture the capabilities in raw format, i.e. specific to the individual methods used, such as CC/PP. The Capability Manager has to be able to understand and to interpret all the different formats and to provide the capabilities in a common format to the Delivery Context Handler. This means, the interface between the gateways and the Capability Manager has to be implemented according to the gateways, i.e. there has to be a protocol adapter for each different kinds of gateway. The interface between the Capability Manager and the context delivery handler uses the XDCL specification (introduced in section 3.5.6) to describe the capabilities in a common format.

The Access Session describes the connection of the user's terminal to the I-centric User Interaction system. Therefore, the capabilities of this connection are grouped according to the Access Session identifier. The gateways provide the captured capability data together with the Access Session identifier to the Capability Manager. Accordingly, the Delivery Context Handler can query the capability set by specifying the according Access Session identifier belonging to the Delivery Context. If the Delivery Context does not have an Access Session, i.e. the user does not have a connection to the I-centric User Interaction system, the Delivery Context Handler does not interrogate the Capability Manager, and the provided Delivery Context data set does not contain any capabilities data.

3.2.8 Access to the Ambient Information System

The Ambient Information System gathers and manages ambient information, which is provided by sensor networks or also by gateways similar to the provisioning of capabilities of the Access Mechanism. Accordingly, the gateways can derive ambient information directly from the terminal and from the connection between terminal and I-centric User Interaction system. For example, the terminal could be equipped with sensors, e.g. to sense temperature or position. The according gateway, which manages the connection to the terminal, is responsible to receive such data from the terminal. These could be provided by standardized concept, such as CC/PP, or also proprietary concepts, which can be specific to the vendor of the terminal device. Accordingly, the gateways have to implement suitable support to gather such ambient information from the terminals.

The gateways store the captured ambient information data to the Ambient Information System as specified in the ambient awareness functions described in section 3.4.4. If possible, the gateways provide the location relation of the ambient information, but at least they have to specify the reference to the Access Session. This is necessary to enable the Ambient Information System to compile a set of relevant ambient information to be provided to the Delivery Context Handler. This means, the Delivery Context Handler can interrogate the Ambient Information System by providing in combination or individually a location reference, user identification, or Access Session identification. The Ambient Information System searches its database according to the given parameters and returns a compiled set of ambient information according to the ambient data structure, which is defined in section 3.4.3.

3.2.9 Access to the Preferences Management System

As described in 3.3.6, the Personalization model defines two interfaces to be provided by the Preferences Management System – the internal and the external interface. The Delivery Context Handler uses the internal interface of the Preferences Management System to obtain the user's preferences. The Preferences Management System itself determines the active User Profile, by evaluating the available Delivery Context data according to the conditions specified by the User Profile's Selection Context.

Accordingly, the Preferences Management System returns the user settings as specified by the active User Profile for the current service. Additionally, the Delivery Context Handler inquires the general user preferences to be included into the Delivery Context. The Preferences Management System returns the preferences as a list of attributes, which can be simply included in the Delivery Context structures as specified in Figure 3-17.

3.2.10 Conclusion

The presented Service Adaptation Framework introduces an approach to realize the concepts of the different models coherently. By using the framework, services can interact with the user according to the objectives of I-centric User Interaction.

The framework specifies a number of functional elements, namely the Capability Manager, the Ambient Information System, the Preferences Management System, the Delivery Context Handler, and the Service Adaptation Function. The specification of these individual components is derived from the respective models. The Service Adaptation Framework specifies the cooperation of these components by defining suitable interface and data exchange formats.

The main component of the framework is the Service Adaptation Function. The presented specification describes detailed the individual steps of the transformation process. This transformation pipeline considers user preferences as well as ambient information whenever necessary. In this way, the user interaction appears personalized and ambient aware to the user.

Furthermore, the Service Adaptation Framework specifies a suitable session concept differentiating between Access Session and Service Session. The Access Session describes the connection of a terminal to the system and the Service Session describes the actual service interaction state. This allows suspending a service usage at any time by ending the terminal connection and hence the corresponding Access Session. The suspended Service Session can be resumed later with a new Access Session. In this way, the user can hand over the interaction to another terminal, possibly changing also the user interface technology, e.g. from WWW-based access to WAP-based access.

The presented concept of the Service Adaptation Framework implements various ideas, which are provided by the models for Personalization, Ambient Awareness, and Generic User Interaction. In the following sections, these models are introduced giving a detailed description of the individual concepts.

3.3 Personalization Model

The Personalization model, presented in the following sections, was developed based on the identified requirements in order to support I-centric User Interaction. After the general introduction, which repeats and refines the individual objectives, the underlying information model to describe and to structure personal preferences is introduced.

Following that, relevant use cases with respect to the support of I-centric User Interaction are discussed. This includes a description of the involved actors. Subsequently, a detailed definition of the resulting functional extent of the Personalization model, including the specification of interfaces and functional elements, is given.

3.3.1 Introduction

The reference model for I-centric Communications defines personalization as a necessary feature to be supported by I-centric services. The personalization of a service means to consider personal preferences of the user for the service usage. Human beings have different demands, different goals, and therefore different preferences. Even after a careful market studies about user demands, a service can only satisfy the demands of a specific subgroup of potential users. By applying suitable personalization concepts, this group can be enlarged and the revenue for the service provider can be increased. Customers prefer personalized services, because they are a more comfortable and user-oriented considering the particular preferences of users. Therefore, the personalization feature depicts an important quality aspect especially in market-segments with strong competition, such as the mobile telecommunication sector.

In order to appear personalized, services have to interpret and to adapt to the user's personal preferences. The Personalization model provides a method to describe personal preferences and specifies the necessary functions for the management of the personalization data. A *Preferences Management System*, which implements the Personalization model, provides suitable APIs through which the preferences data can be accessed and managed. The specified functions reflect the requirement to support different access schemas ensuring privacy and security.

One prerequisite in order to enable personalization is the *identification* of the user. Based on the identification, the corresponding personal preferences can be queried from the preferences storage. Here, the term user does not necessarily have to refer to an individual person. Instead, it can also refer to a user group. It depends on the applied personalization method, whether a service can adapt to the preferences of an individual or to the preferences of a group of individuals. For example, there could be two kinds of users: novices and professional users. Both have different preferences for using a certain service. Here, the identification process should determine the group, to which a user belongs, but does not necessarily need to identify the individual user in order to query the correct preferences.

When developing a service, the service developers have to identify the parameters, which can be used by the services for the personalization. These parameters can be expressed as name/value pairs, with a well-defined name and well-defined possible values. This means, the services provide a list of parameters that have to be managed. On the one hand, the services could manage these parameters for each user themselves. This would require that services would have to implement the complete Personalization model. On the other hand, there are a number of reasons, which motivate a more centralized approach, i.e. multiple services share personalization data and related functions:

- Each service does not have to implement an own Personalization model. Instead, services can easily use existing implementation of the Personalization model. This saves development effort.
- Services use the same personalization model, which makes the usage of the services more transparent to the user. Different personalization models would provide different features, usage, and qualities, which in turn would deter the users from using the services.

- User preferences already obtained from the user can be made available to all services under consideration of relevant security aspects.
- When the personalization model is extended by new function or features, all services can profit from them immediately.

Figure 3-21 shows the personalization function with a Preferences Management System providing a suitable programming interface to the services in order to enable the common usage. As stated above, personalization is a feature, which is to be implemented by the services. The Personalization model supports the services in this by providing the necessary base functions.

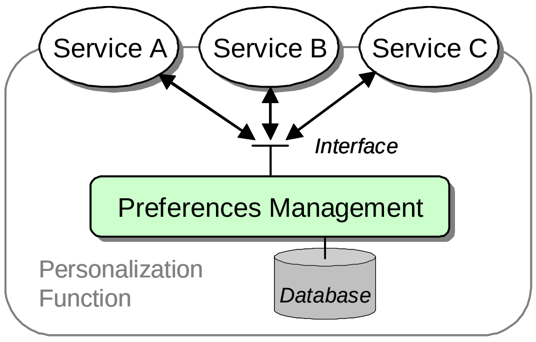


Figure 3-21: Central Personalization Model Approach

According to the I-centric User Interaction idea, the preferences of the users are not only relevant to the services. They should be considered in the realization of the user interaction, likewise. Beside the preferences with regard to the service behavior, the users might have preferences with regard to the service appearance, i.e. the user interface. The Personalization model should support the management of both kinds of preferences coherently.

Furthermore, preference data can be distributed. Today's terminals are capable devices, able to store data and therefore to support the personalization. For example, most mobile phones provide several profiles, which prescribe the behavior and the appearance of the phone, e.g. ring tone, loudness, menu size, etc. Additionally, such profiles also can specify preferences, which can be likewise relevant to a service, which is not executed on the device directly, e.g. language preferences or presence information. That is why the Personalization model should support the management of distributed personalization data. A solely centralized approach will not be sufficient for future application. This would require a permanent connection to the Preferences Management System, which is not possible and unsatisfactory especially in mobile systems.

Furthermore, from the user's viewpoint the Personalization model itself has to be simple and transparent. Although, users might profit from the personalized behavior of the system, they do not want to feel the technical details and of the realization and will not accept any additional efforts in the usage. The system must not rely on experienced users or expect an increased training effort. This also regards to the sensible dealing with the user data. The system has to ensure the users' privacy in order to be accepted as a being trustworthy. Beside the technical concerns, these issues are the most relevant and critical ones in today's systems.

3.3.2 Overview

As stated above, the identification of the user, i.e. individual or group, is the precondition to allow personalization. The preferences, to be considered in the service provisioning, are always assigned to an according user identifier. Based on the identification, a Preferences Management System provides the available preference data. There has to be a common identification and authorization schema through all the elements of an I-centric User Interaction system.

The Personalization model, as presented here, provides a concept to describe and to structure user preferences, i.e. an information model for personalization data.[18] Furthermore, it defines necessary function in order to specify, access, and evaluate preference data, i.e. the interface to the personalization functions. The actual personalization process has to be performed in the service provisioning directly, i.e. in the service logic as well as in the interaction system such as the I-centric User Interaction system. In order to specify a concept to describe personal preferences allowing a suitable personalization, some assumptions have to be considered:

- Users have different preferences for different situations, e.g. devices, locations
- Users have different preferences according to the time
- Users should provide their preferences as comfortable and easy as possible, i.e. without or at least with less as possible explicit interrogation
- Preferences can be derived from evaluation of recent service usage, which depicts an alternative to the explicit stating
- Services can also gather preferences from the user during the service usage
- Services have to consider the preferences of the service user and need therefore suitable access to the preference data
- The interaction of the user with the service can also consider preferences for the appearance or presentation
- From the interaction between user and service also preferences can be derived and be used further on as the user’s preference

In order to realize a personalization system supporting these assumptions, the underlying Personalization model has to provide suitable concepts and solutions. Accordingly, these have to comprise the specification of:

- A general information model to describe user preferences
- Necessary functions to gather, to manage, and to provide user preferences including necessary exchange formats

The technical specification [3GPP-TS22.121] defines models and mechanisms essential for the provisioning of personalized services in the (Virtual) Home Environment. Accordingly, a Personal Service Environment (PSE) contains all information with regard to the provisioning and presentation of subscribed and possibly personalized services provided in the VHE towards the user. Therefore, the personal service environment maintains a set of one or more User Profiles. According to the specification [3GPP-TS22.121], a User Profile is defined as follows:

A *User Profile* is a set of information necessary to provide a user with a consistent, personalized service environment, irrespective of the user location or the terminal used (within the limitations of the terminal and the serving network). The user can define one or more User Profiles according to the user’s needs. The user’s home environment manages the User Profile(s).

The logical VHE role model, defined in [3GPP-TS22.121] (Figure 3-22), describes the relationships between the various components and roles involved in the realization of a VHE according to 3GPP.

[18] Initial approaches and concepts are described in the diploma thesis [RÄCK].

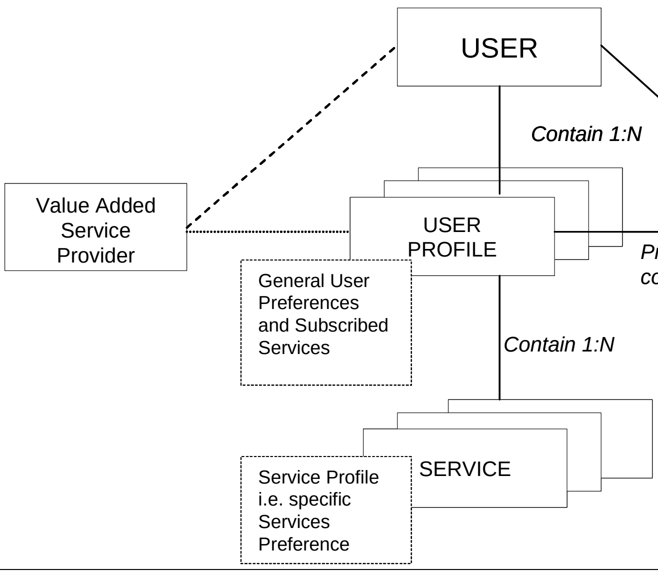

USER
Contain 1:N
Value Added Service Provider
USER PROFILE
Home Environment
Provided and controlled by
General User Preferences and Subscribed Services
N:N
Contain 1:N
HE VASP
SERVICE
Service Profile i.e. specific Services Preference

Figure 3-22: Logical VHE Role Model

The model consists of the home environment, the user, one or more Value Added Service Providers (VASPs), one or more Home Environment-Value Added Service Providers (HE-VASPs), and one or more User Profiles. The home environment should allow the management of the User Profiles by the user, e.g. activate, modify, deactivate, etc., the access to specific User Profiles by authorized HE-VASPs, as well as controlled and limited access to specific User Profiles by VASPs, e.g. for general user preferences and subscribed services information.

The User Profile is a combination of information used for the customization of the services via the home environment system, e.g. user preferences according to the general presentation of services, and preferences associated with subscribed services, i.e. service related preferences. The following requirements were taken into account for the development of the presented Personalization model:

- **Classification**: Two types of data should be supported by the User Profile. There are so-called general preferences, which are independent of the subscribed services, e.g. user interface preferences, and there are service-specific preferences, which are defined by the individual types of services.
- **Management, Provisioning and Access**: It should be possible to create, delete, update, and read the User Profile data.
- **Selection and Activation**: The user or a home environment service provider should be able to define a Default User Profile or the criteria for selecting and activating a certain User Profile dynamically.
- **Location and Distribution**: The distribution of User Profiles or certain parts between different entities should be possible. I.e. the home environment should be able to hold a backup copy of the data stored within the user equipment if required.
- **Synchronization**: Due to the distributed nature of the User Profile content, there should be a mechanism supporting the synchronization and recovery of this information.
- **Default Policies**: Different policies regarding the way the User Profile data can be modified should be support by the home environment.
- **Security and Privacy**: Access to the data stored in User Profiles should only be permitted in an authorized and secure manner. Furthermore, the transfer of User Profile data should take place in a secure way.
- **Format**: A common syntax and semantics for a profile language are needed in order to allow standardized access, interoperability, and synchronization of the User Profile data.

Based on these requirements and by considering the user's point of view, the following top-level criteria for a Personalization model can be derived as follows:

1. **Management and Grouping of Preferences:** It should be possible to create, read, update, or delete preferences. Preferences belonging to the same service should be kept together, i.e. there are different sets of preferences each belonging to a particular service.
2. **Security and Privacy:** Secured and authorized access to the stored information should be assured. Additionally, all transfer of user information and preferences should be secured as well.
3. **Implementation Interoperability**: It should be possible for different implementations of the same Personalization model to exchange their stored information even if completely different technologies have been used for the implementation.

The Personalization model for I-centric User Interaction systems therefore includes an information model (section 3.3.3) and a functional specification (section 3.3.5). These specifications have to consider the different roles of the components, which use the personalization system. Accordingly, their access rights to the preference data as well as to the personalization function have to be specified. This is necessary in order to realize a trustworthy, reliable, and liable personalization system. From the assumptions, presented at the beginning of this section, as well as from the VHE role model as described in [3GPP-TS22.121], the following actors can be identified:

- The service user
- The service
- The user interface, i.e. the Service Adaptation Function of the Service Adaptation Framework (section 3.2.5)

The user of a service specifies the preferences for the service behavior and service appearance. The user can be an individual or a group of individuals with the same preferences depending on the target of the personalization. The preferences have to be gathered from the users and to be provided to the Preferences Management System. The services on the one hand use the available preferences in order to adapt their behavior, but on the other hand, they also can provide preferences, e.g. the user could specify preferences during the service usage and the service stores the information as the user's preference. In order to personalize the appearance the Service Adaptation Function of an I-centric User Interaction system has to consider user preferences, too. In this way, the adaptation of the user interaction can be personalized, e.g. arrangement of content, languages, layouts, styles, etc. The uses case diagram shown in Figure 3-23 presents these high-level use cases.

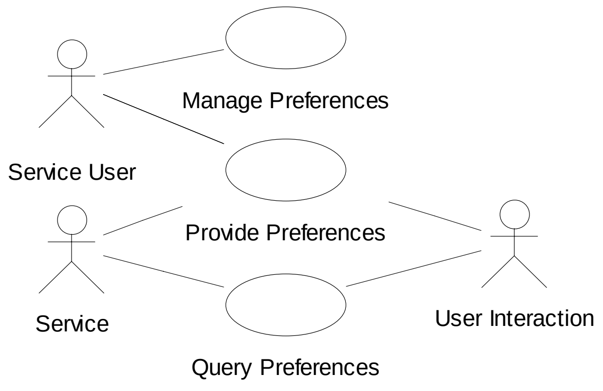


Figure 3-23: General Personalization Use Cases

The individual use cases are explained in the following sections detailed. They depict the general base, on which the Personalization model was developed. After the introduction of the concept to describe and to structure user preferences (section 3.3.3) and the functional specification for the management of preference data (section 3.3.5), the last section finally discusses some security and privacy concerns (section 3.3.8), which have to be taken into account for the implementation of the proposed Personalization model.

3.3.2.1 Use Case: Provide Preferences

Obviously, the users have to provide their preferences to be considered in the service provisioning. Nevertheless, the procurement of preferences data should be user-friendly. This means, the user should not be burdened with long-lasting procedures to specify all the necessary data. Instead, services should consider preferences if they are available, but should not rely on their availability. The individual preferences can be derived from the usage of a service *explicitly* and *implicitly*.

Explicitly gathering means to interrogate the users about their preferences, preferably at the time when the information is needed, i.e. to provide options such as ‘change language to...’. Once the user has provided such explicit input, this information can be used further on and the user has not to be asked again.

Implicitly gathering means to watch the user’s service usage carefully. By providing reasonable interaction, meaningful conclusions can be drawn from the interaction. This requires some self-learning capabilities, which have to evaluate all relevant data. For example, if the user selects always a certain news category in a news service and ignores the other categories available, it can be derived that the user has an intensified interest in news belonging to this category. The service can than adapt to this user accordingly by presenting such news at first. This requires a sensible evaluation of the user’s feedback, possibly applying some rating mechanism.

Of course, the user should always have the possibility to change the preferences afterwards. Furthermore, specific preferences can be preset by the system. If the user database is extensive enough, ‘pattern-matching’ algorithms can derive probable preferences from other users. In this way, certain user groups could be identified, such as teenagers or business users, depending on general information about the user, e.g. age, origin, sex, profession, etc.

3.3.2.2 Use Case: Query preferences

The services themselves but also the interaction between user and service, i.e. the Service Adaptation Function, consider the user’s preferences during their execution. Therefore, the personalization system has to provide the relevant preference data to these components, i.e. the services have to retrieve these data from the personalization system. This means to offer the relevant information in a suitable form. The relevance is dependent on the context, in which the service usage takes place. The user can have specified some conditions for the service behavior, e.g. timely restrictions or location dependencies. These conditions have to be evaluated according to available contextual information; and the corresponding the preferences have to be provided to services and to the Service Adaptation Function.

For the retrieval, i.e. the exchange, suitable interfaces have to be provided and according data formats have to be specified, which enable an efficient information exchange. The components, which process the preferences, should obtain the prepared data in a comfortable arrangement, which enables an easy integration and hides the complexity of the underlying Personalization model.

3.3.2.3 Use Case: Manage Preferences

In order to specify the conditions, under which certain preferences shall be used, the users have to manage their personalization data. This covers the creation and deletion of profiles, the specification of corresponding conditions, as well as the declaration of the general user preferences.

Such functions should be provided to the user in a comfortable and easy-to-use interface. However, the personalization system should not necessarily expect these actions from the users. Instead, it should provide them as an add-on functionality to exhaust to full potential of the provided personalization features.

3.3.3 Structured Description of Personal Preferences

Looking at the human way to state personal preferences, it can be determined that all relevant information can be modeled as simple key/value pairs. Whereas the actual values can be either character strings associated with their appropriate keys or collections of character strings associated with their appropriate keys as well. These key/value pairs are named *Attributes*. Additionally, the preferences are usually valid for a particular situation meaning that preferences can be different in different situations. This means, there are conditions, here denoted as the *Selection Context*, which specify the preferences to be used. The information model as shown in Figure 3-24 was developed to structure and describe personal preferences and to realize the required personalization features.

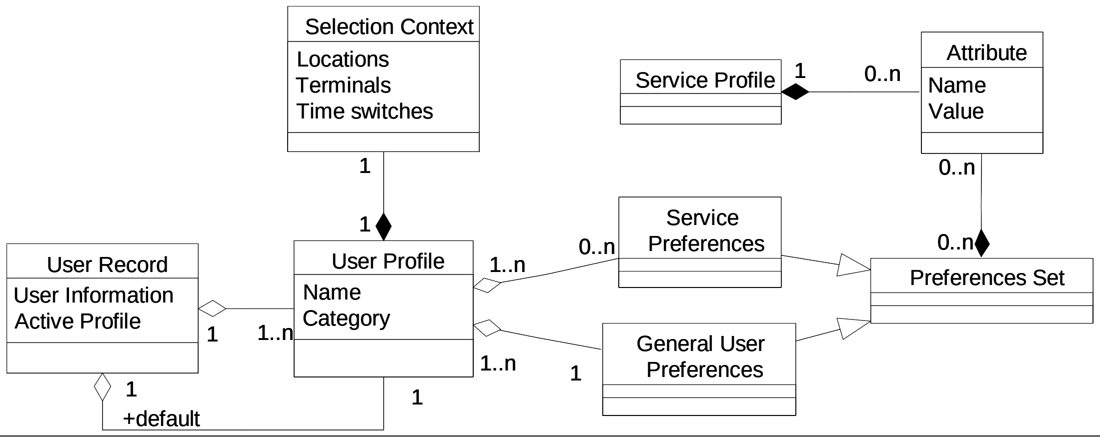


Figure 3-24: Information Model to describe Personalization Data

In this model, each user (or user group) is represented as a *User Record* containing general information, such as name, etc. The user can have a number of *User Profiles* characterized with according conditions, i.e. the *Selection Context*. The User Profiles should have user given reasonable names, such as ‘at home’ or ‘during office hours’. There is one special User Profile, denoted as the *Default User Profile*, which is the initial profile and which cannot be deleted. Each User Profile consists of a number of preferences sets, which contains several Attributes composed according to the service relationship.

The preferences sets are divided into *general* and *service related* preferences. The actual preference data are stored as Attributes in a preference set. There are default values for each preference defined by the service profile, which can be overwritten in the specific preferences set. Figure 3-25 shows an example, how personalization data can be structured according to the information model.

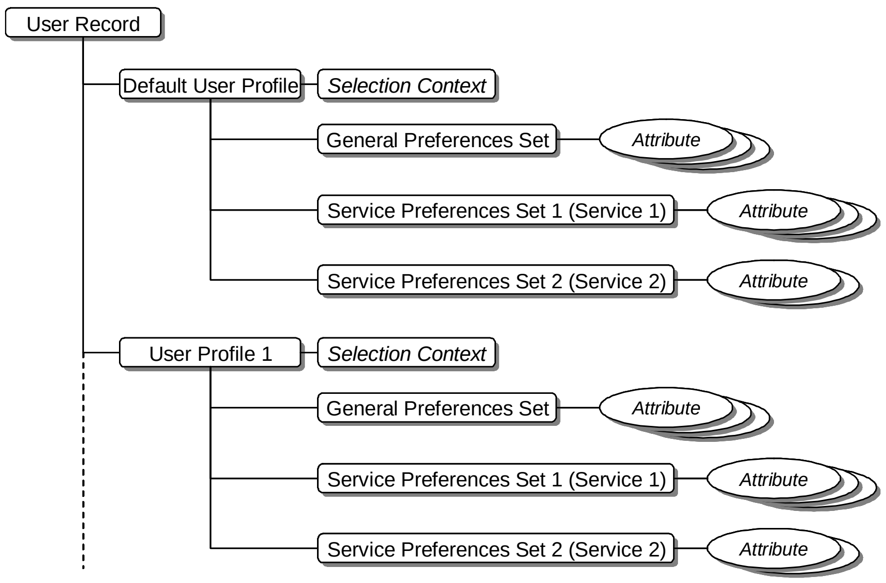


Figure 3-25: Example Structure of Personalization Information

The following sections introduce detailed the individual elements of the personalization information model.

3.3.3.1 User Record

The User Record stores general personal user information, e.g. name, contact data, demographic data, etc. In this context, the User Record represents an individual user or a group of users. Each record maintains a set of profiles belonging to the user, i.e. the User Profiles.

3.3.3.2 User Profiles

User profiles group the general preferences set and the service preferences sets of each user. Each user has exactly one default profile, which has to be used, if there is no other effective User Profile to be used instead. This profile cannot be deleted. It also contains default values for Attributes, which are not specified (overwritten) in the other User Profiles[19]. Each profile belongs to a category and refers to different preferences sets including one general preferences set and several service preferences sets.

By referring to a Selection Context, each User Profile has a condition set, which has to be valid in order to use the preferences assigned to this profile. The conditions have to be unambiguous in order to determine the respective User Profile, which is to be used in a certain situation. This User Profile is denoted as the effective User Profile. If there are multiple User Profiles with valid conditions, priorities provided by the user have to be evaluated. Furthermore, the user can also manually select a User Profile, which then is to be used regardless of the Selection Context. This means i.e. at any given time, there is exactly one User Profile, which contains the current user preferences for a given situation.

The term profile is broadly used in many different situations and applications. Generally, it describes a specific collection of information to be used in a certain application context. By using profiles, it is possible to define the activation context for number of preferences at once.

3.3.3.3 Selection Context

The Selection Context describes the activation conditions for User Profiles. Each User Profile can be characterized with such conditions, which means that a User Profile shall only be used

[19] See section 3.3.3.9 for examples

when all the conditions are fulfilled. Examples of such conditions can be the time, location, used terminal, etc. This is related to the Ambient Awareness model (introduced in section 3.4) and the Delivery Context (introduced in 3.5.6). Accordingly, the defined conditions have to be compared to available contextual information, which are provided by the Delivery Context. In this way, users can specify different preferences for different contexts. The special Default User Profile is always selected, if there are no other User Profiles with valid Selection Contexts. This means, the Selection Context of the Default User Profile cannot be changed.

The attributes of the Selection Context are Boolean expressions, which can be evaluated. If all attributes, i.e. conditions, are true, the Selection Context is valid. The information model does not predefine specific types of conditions, because these are implementation dependent, i.e. what kinds of condition data are available in an actual system. Some possible and reasonable examples are time switches, location references, or terminal identifier.

3.3.3.4 Preferences Set

Attributes, which are related[20] to each other, are combined in a preferences set. Therefore, the set depicts a list of Attributes. There are two distinct classes for preferences sets – the general user preferences set and the service preferences set.

3.3.3.5 General User Preferences Set

The general user preferences set contains common user preferences, e.g. language, interests, etc. This means, all preferences are stored in this set, which are not specific to certain services but are relevant to all services. All services share this preferences set and can therefore access these Attributes.

There is one general user preferences set per User Profile. The general user preferences set of the Default User Profile depicts the default general user preferences set. This means, if Attributes are not specified in the general user preferences set in other User Profiles, the Attributes from the default set are used instead. This enables the user to specify the preferred language only once without needing to specify this again for each profile. However, the user can specify a different language in another User Profile, which is used, when this User Profile is active.

3.3.3.6 Service Preferences Set

A service preferences set contains the specific preferences for each service. This means, there is exactly one service preference set for each service in every User Profile. The services define their own set of preferences (Attributes) and have to provide the default values by the service profile. Each service preferences set describes the user related settings of the corresponding service, e.g. service specific Attributes. The set does not necessarily have to contain all Attributes, which are predefined in the service profile. The service preferences set of a User Profile refers to the default service preferences set, i.e. the one of the Default User Profile, which in turn refers to the service profile. Therefore, each set depict only the difference to the default service preferences set, which depicts only the difference to service profile. This means, if a specific attribute is not specified in the service preference set, automatically the corresponding attribute of the default service preferences set or of the service profile is to be used instead. Table 3-3 contains an according example of the reference mechanism.

For each service, there can be multiple preferences sets assigned to different profiles. According to the currently active profile, the corresponding service preferences set is be considered in the service usage.

[20] See following sections describing the possible relationships

3.3.3.7 Service Profile

The service profile describes all Attributes, by which the service behavior can be personalized. It contains all Attributes with default values. If a service attribute is not specified in a service preferences set, i.e. not specified by the user, its according value is taken from the service profile, i.e. the default setting provided by the service is used instead. Thereby, users do not have to specify each preference in advance for each service.

For example, a weather service could provide the weather forecast in summarized or in detailed form. The service specifies that the default presentation shall be the summarized form, i.e. it defines the default value accordingly. If the user did not provide a different setting, i.e. there is no entry in the current and in the user's Default User Profile, the default value of the service profile is used. If the has user specified a personal preference for this attribute, i.e. there is an entry in the service preference set, this value is used instead. Table 3-3 contains a usage example describing this approach.

3.3.3.8 Attributes

The Attributes are key-value pairs containing information utilized by service providers to customize their services. These depict the actual elements to control the personalization. The types of the Attributes are dependent of the respective service, e.g. plain strings, numbers, alternatives, etc. In the presented Personalization model, they are defined as character strings. Non-string values can be stored in a serialized representation, i.e. as 'stringified objects'. Because only the services provide the actual values as specified by the user during the service usage, the service have to take care about the reasonability of the values. The Attributes are always organized in preferences sets.

3.3.3.9 Usage of the Information Model

As described above, the proposed information model to describe personalization data uses a hierarchical information structure. For example, preference sets are actually updates to the corresponding default preference set, which is in turn an update of the according service profile. The updates overwrite the Attributes of the origin data set. Table 3-3 shows an example explaining the dependencies of the different data sets more detailed.

		Default Service Profile	***Attribute 1***	***Attribute 2***	***Attribute 3***
		SP1	value1	value2	value3
		SP2	value5	value6	–
User Record	***User Profiles***	***Preferences Sets***	***Attribute 1***	***Attribute 2***	***Attribute 3***
UR1	UP0 (Default User Profile)	GPS	value7	value8	–
		SPS1	value9	value10	– {value3}
		SPS2	value11	– {value6}	–
	UP1	GPS	value12	– {value8}	–
		SPS1	– {value9}	– {value10}	– {value3}
		SPS2	value13	– {value6}	–
UR2...	UP0...	...	...	...	...

Table 3-3: Usage of User Profiles

In the given example, there are two services, which provide corresponding service profiles (SP1 and SP2). SP1 defines three Attributes and provides the default values for each (value1-value3).

SP2 on the other hand only specifies two Attributes. The example does not use reasonable names or values, but uses figurative samples instead.

Furthermore, the example defines one User Record (UR1) detailed. This User Record has two User Profiles (UP0 and UP1). Each User Profile specifies one general preference set (GPS) and several service preference sets (SPS), one for each service, i.e. SPS1 for service 1 and SPS2 for service 2.

UP0 depicts the Default User Profile. This means, UP0 has to be used if there is no other effective User Profile. In addition, the attribute values specified in the preferences sets of UP0 (GPS, SPS1, and SPS2) have to be used if they are not specified in the other User Profiles, e.g. attribute1 in SPS1 of UP1. If Attributes are not specified in the Default User Profile, the original values from the service profiles are to be used, e.g. Attribute3 in SPS1 of UP1 and UP0.

3.3.3.10 Preference Description Language

This information model describes how the preferences data are structured internally, i.e. within the preference management function. This means, it represents the data model for the underlying storage system. However, a format for the exchange of preference data is needed, too. Today's personalized services do not provide standardized preferences storage mechanisms. This results in a rather undesirable situation from the user's perspective. Users are required to keep many different personal accounts when using multiple services. This situation severely hampers the wide acceptance of personalized services. Mechanisms to share and to exchange personalization data is required to provide a seamlessly personalized service environment to the user.

The Microsoft Passport [MSPASSPORT] and the Liberty Alliance Project [LIBERTYA] represent no real personalization concepts in general, but they address the service interoperability issue (in the field of authentication). Both concepts define mechanisms for online identification, which allows individuals to carry their identity from site to site. They propose a kind of 'single sign-on' technology that enables users to log on at one web site, i.e. at the home environment provider, creating an authenticated session, which allows the usage of services, which are provided by other participating web sites, i.e. by value-added service providers.

These approaches are primarily focused on facilitating B2C relationships over the Internet by simplifying the process of establishing a global identity. This identity can be shared among different participating sites. They do not address personalization directly although Microsoft's Passport allows users to share some information among participating sites, like e-mail addresses, phone numbers, or preferred languages, etc.

However, to enable personalization data exchange and shared usage, a common preferences description language and according protocols are required. This will results in an improved service interoperability and thus in an improved long-term customer relationship. Fink et al. state in [FINK] that this approach offers significant advantages with respect to the:

- "Management and retrieval of (user-related) information, which is compliant with widely established standards allowing for easy integration."
- "Addition of new information types to the set of predefined types, which allows for flexibility as sites and personalization goals evolve."
- "Distribution of information across a network leading to better performance, scalability, availability, and reliability."

According to [WERNEMAN], there is already a great variety of description methods, which are used within the 3GPP specifications. These methods include ASN.1, textual, CC/PP, UML/IDL, and XML schemas.

The 3GPP technical specifications [3GPP-TS22.240] and [3GPP-TS23.241] introduce the Generic User Profile (GUP) and the GUP Data Definition Framework (DDF), which are likely to be used as the preferred description methods within further 3GPP specifications. GUP uses the DDF language for the profile description. DDF is based on XML.

GUP DDF offers a flexible and extensible basis for the description of personalization information. Nevertheless, it has one big disadvantage compared to RDF/XML. RDF and the RDF/XML syntax are already widely accepted in the industry. The CC/PP framework, which is based on RDF/XML, is expected to be supported by future mobile phones and Internet appliances. The support for GUP DDF is inferior compared to the support for RDF/XML.

The Platform for Privacy Preferences Project (P3P) enables WWW sites to express their privacy practices in a standard format that can be retrieved automatically and interpreted easily by user agents. P3P user agents will allow users to be informed of site practices (in both machine- and human-readable formats) and to automate decision-making based on these practices when appropriate. Thus, users need not read the privacy policies at every site they visit. [P3P]

Although P3P provides a technical mechanism for ensuring that users can be informed about privacy policies before they release personal information, it does not provide a technical mechanism for making sure sites act according to their policies. Products implementing this specification may provide some assistance in that regard, but that is up to specific implementations and outside the scope of this specification. However, P3P is complementary to laws and self-regulatory programs that can provide enforcement mechanisms. In addition, P3P does not include mechanisms for transferring data or for securing personal data in transit or storage. P3P may be built into tools designed to facilitate data transfer. These tools should include appropriate security safeguards. [P3P]

[APPEL] complements the P3P1.0 specification [P3P] by specifying a language for describing collections of preferences regarding P3P policies between P3P agents. Using this language, users can express their preferences in a set of preference-rules (called a rule set), which can then be used by the user agent to make automated or semi-automated decisions regarding the acceptability of machine-readable privacy policies from P3P enabled WWW sites

Summarizing, it can be determined that there are a number of different approaches for the exchange of personalization data. Some are already disappearing, because they did not gain foothold in commercial applications, whereas others are still under development and their future importance and acceptance is open. Fundamentally, the individual approaches are related to specific application platforms, i.e. APPEL and P3P for the WWW, CC/PP for the mobile sector. However, they depict only generic concepts, which cannot be used as they are. For the time being, there is still no suitable uniform description language for the exchange of personalization, but there a lot of activities, which depicts the relevance and the need.

For the Personalization model presented in this thesis, this means that the individual information objects need to be mapped to suitable representations according to the target exchange format. This thesis does not propose an additional, new approach for such exchange formats, but encourage the developers to look for available and widely used specifications. According to the selected exchange language suitable *adapters* with corresponding *transformation rules* have to be specified and to be implemented. A possible approach is depicted in Figure 3-26.

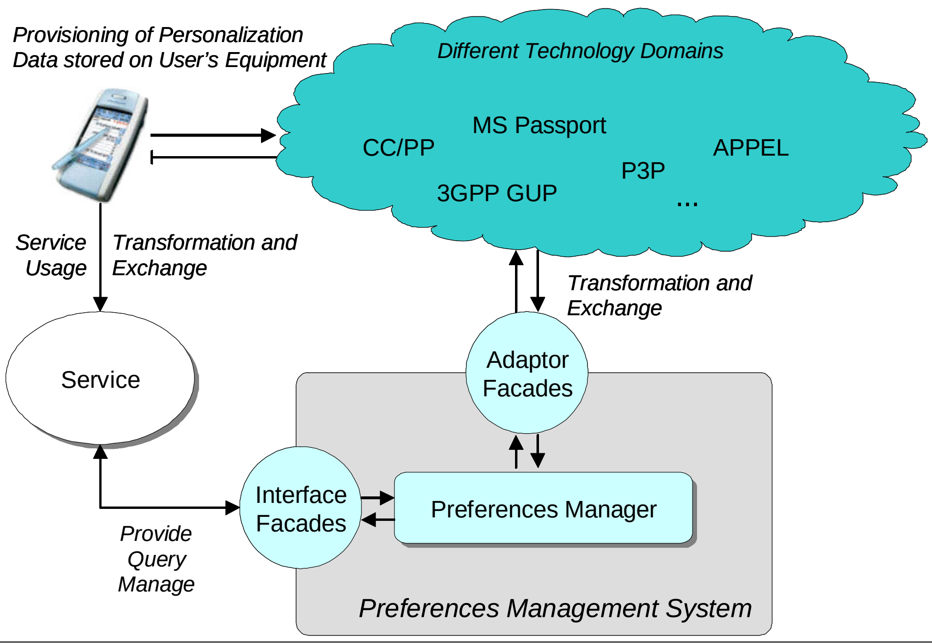

Provisioning of Personalization Data stored on User's Equipment
Different Technology Domains
MS Passport
CC/PP
APPEL
P3P
3GPP GUP
...
Service Usage
Transformation and Exchange
Transformation and Exchange
Service
Adaptor Facades
Interface Facades
Preferences Manager
Provide Query Manage
Preferences Management System

Figure 3-26: Using an Adapter to transform Personalization Data to external Description Systems

The transformation rules have to be defined in accordance to the selected description languages. The rules have to specify on the one hand a syntactical mapping, which can be defined simply, but on the other hand, they have to provide a logical mapping of the actual data elements. If the information models behind are not compatible, this is only partially possible. For example in CC/PP, the User Record, the general user preferences set, and the service preferences sets introduced by the Personalization model in this thesis, would need an according CC/PP representation in addition to the representation as defined in the sections above. The defaults preferences sets could be simply represented as a CC/PP defaults profile, whereas the other preferences sets could be represented as CC/PP profiles. However, CC/PP is still a general specification. For the containment of complex personalization data, an according module would have to be defined additionally.

3.3.4 Usage Scenarios for the Personalization Model

In order to derive the necessary functions of the Personalization model, the possible usage scenarios have to be evaluated. First, the different actors of the usage scenarios are identified and described in section 3.3.4.1. Then, according use cases with regard to the personalization functions are introduced in section 3.3.4.2. These use case are the base for the definition of the core personalization functions, which are introduced and described in section 3.3.5.

3.3.4.1 Relevant Actors

Business and technical separations lead to the definition of different business roles. The previous sections already covered the main actors implicitly. This section introduces them explicitly. Five business roles can be distinguished:

The **User** consumes the services and is main business target. In terms of personalization, the user can be an individual or describe a group of individuals with certain relationship (same preferences).

The **Home Environment Provider** (HE) maintains user relationship and enables the user to access to services. The HE provider supports a number of functions centrally, which can be used by services in order behave I-centric, i.e. to adapt to user's preferences and current context.

The **Home Environment – Value Added Service Provider** (HE-VASP) offers different kinds of value-added services to users and uses the central functions provided by the HE provider.

These services are interior services integrated into the HE platform, e.g. a user preference management service.

The **Value Added Service Provider** (VASP) can offer different kinds of value-added services to users using the central functions provided by the HE provider. These services are exterior services. The HE-VASP can be a reseller of the services provided by a VASP. However, there can be also separate business relationships between user and the VASP, i.e. usage contract and billing.

The relationships between the different actors are shown in Figure 3-27.

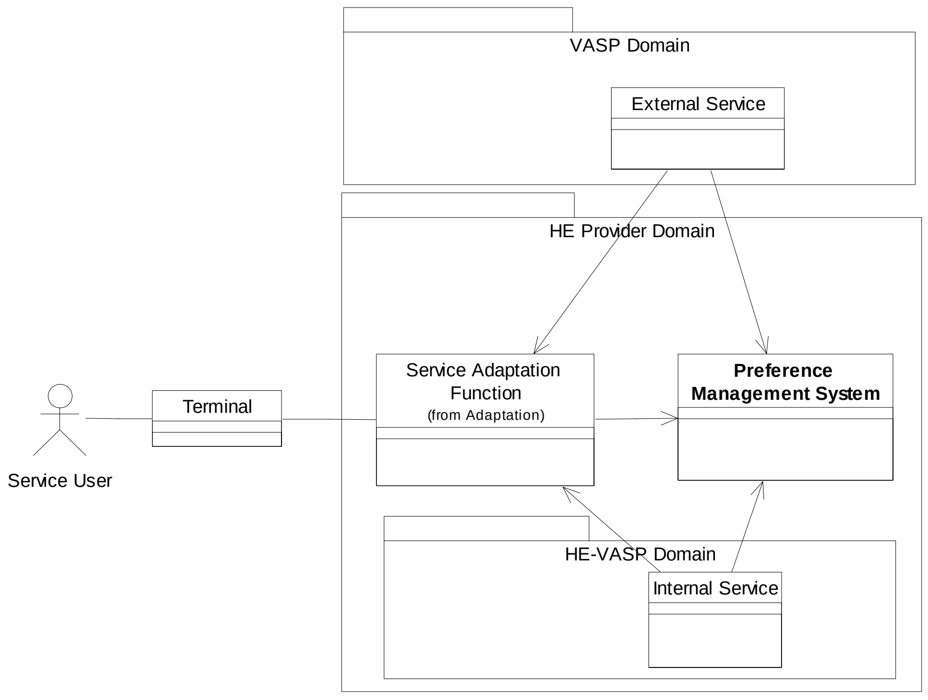


Figure 3-27: Relationships between the different Actors and Domains

Via their terminals users access services provided to them by the HE-VASP and by the VASP. The services are separated into internal (HE-VASP) and external services (VASP). All service providers shall be able to personalize their services with the help of personalization function provided in the home environment. Components, which belong to the home environment, such as the Service Adaptation Function, are treated like the HE-VASP actor. Service users use the Personalization model indirectly by according services provided by their home provider, such as a special service to manage their own account, e.g. to create and delete profiles, specify the time switches, etc. On the other side, VASPs do not manage the complete personalization data. They just need to gather the needed data to personalize their services. Actually, it has to be ensured that VASP's cannot access and modify data, which does not belong to their services. They only are allowed to access their service preferences.

Because of their different role, the different actors should have different access rights to the personalization data and personalization functions Whereas HE-VASP can manage personalization data without any restriction, the VASP should only have a limited access to the personalization function, i.e. to read and update only the user's preferences, which belong to the particular service. Therefore, the roles of the different actors can be summarized to:

- The *internal* access, and
- The *external* access

These different access roles are used in the following for the derivation of the personalization functions and hence the according APIs.

3.3.4.2 Personalization Model Use Cases

In order to derive the functional specification of the Personalization model, the required features have to be determined before. These required features can be easily identified in a top-down approach by analyzing what the different involved components need. They can be described by several use cases, which depict different viewpoints for the identification of required functions.

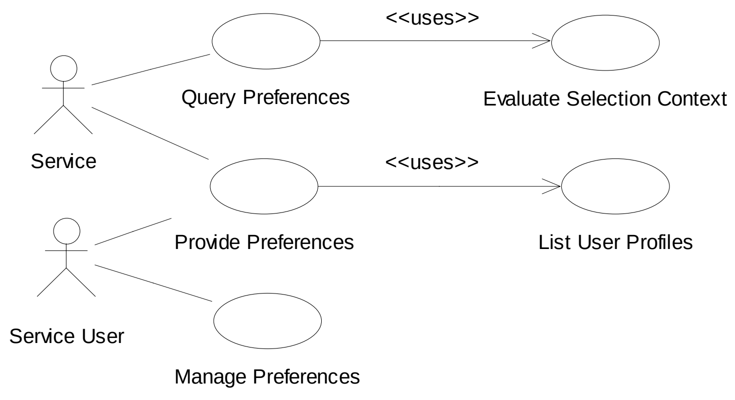


Figure 3-28: Refined Personalization Use Cases

As shown in Figure 3-28 there are three main use cases to be supported by the personalization system:

- Provide preferences
- Query preferences
- Manage preferences

The services use the first two uses cases in order to appear personalized, whereas the last use case describes the administration of the preference data by the user. This should be implemented by a particular service, which for instance enables the user to create profiles and specify the Selection Context for each.

Provide Preferences

When a service is registered, i.e. deployed, it has to provide the default service preferences set. This set describes all the Attributes and their default values to be used further on. If the user does not have specified any personal preferences for this service, these default preferences are considered in the service usage. Because the actual meaning of each attribute is only known to the service, only the respective service can provide reasonable data for them. This means, the services have to enable the user to specify the setting of the preferences and to store them to the preference database. Because service preference sets are always linked to a User Profile, the service has to ask the user, in which profile the selected preferences should be used. This can be a specific profile, but can be also the user’s default service profile. Furthermore, the preferences should be transmitted in a simple structure, for instance in a list of Attributes.

The following required functions can derived accordingly:

- To find out the profiles created by the user (needed to store the preferences)
- To store preferences according to a profile

These functions are used by the service as shown in Figure 3-29.

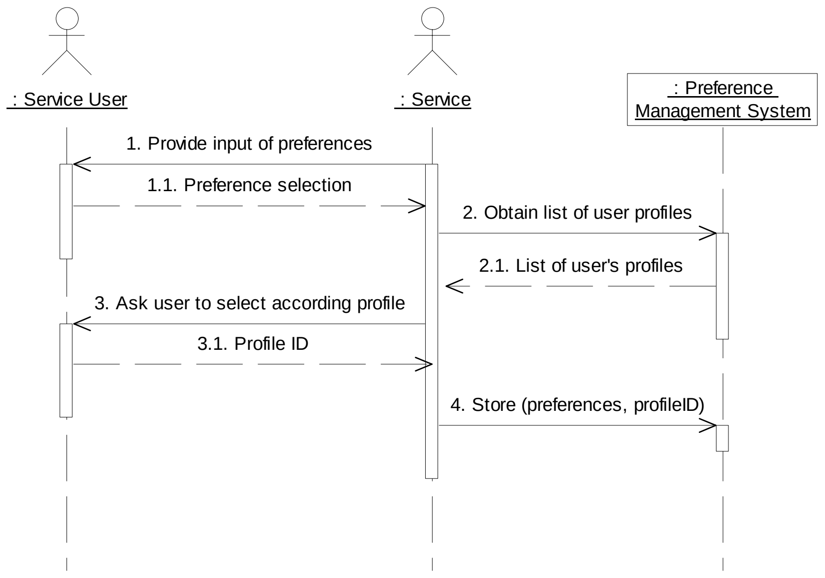

: Service User
: Service
: Preference Management System
1. Provide input of preferences
1.1. Preference selection
2. Obtain list of user profiles
2.1. List of user's profiles
3. Ask user to select according profile
3.1. Profile ID
4. Store (preferences, profileID)

Figure 3-29: Provide Preferences Sequence Diagram

First (step 1), the service incorporates an option to specify a preference setting in the service's user interface, such as 'Select other language...' for example. If the user provides a settings by selecting an option (step 1.1), the service can just use this value for the current session, but can also store the specific settings to the Preferences Management System for future use. If the settings are to be stored, the service first has to obtain the list of the user's profile (step 2) to provide this selection to the user (step 3). This is necessary, because service preferences are always linked to a specific User Profile.

The user selects one User Profile (step 3.1), which can be the user's Default User Profile or any derived User Profile. Finally, the service can store the initially provided preference setting in the selected User Profile to the Preferences Management System (step 4).

Query Preferences

In order to appear personalized, the services as well as the user interaction system have to obtain the preference data, i.e. to retrieve these data from the personalization system. Because the preferences are assigned to respective profiles, which can have a Selection Context assigned, it has to be evaluated first, which profile is currently *effective*, i.e. the active profile. This means, the Selection Contexts of each profiles are assessed and the according preference set is returned.

The evaluation function accesses necessary data, which partially are provided by the Delivery Context, such as terminal type, location, etc. The preferences, which are to be returned, should be described in a simple structure, i.e. just as a list of those Attributes, which the user has changed. This means default Attributes do not necessarily have to be transmitted, because the service knows them already. Beside the service preferences, also the general user preferences and data from the user can be interrogated.

The following required functions can derived accordingly:

- To determine the currently effective User Profile, if no profile is activated explicitly
- To retrieve the user's service preferences assigned to this profile
- To retrieve User Record data
- To retrieve user's general preferences

These functions are used by the service as shown in Figure 3-30.

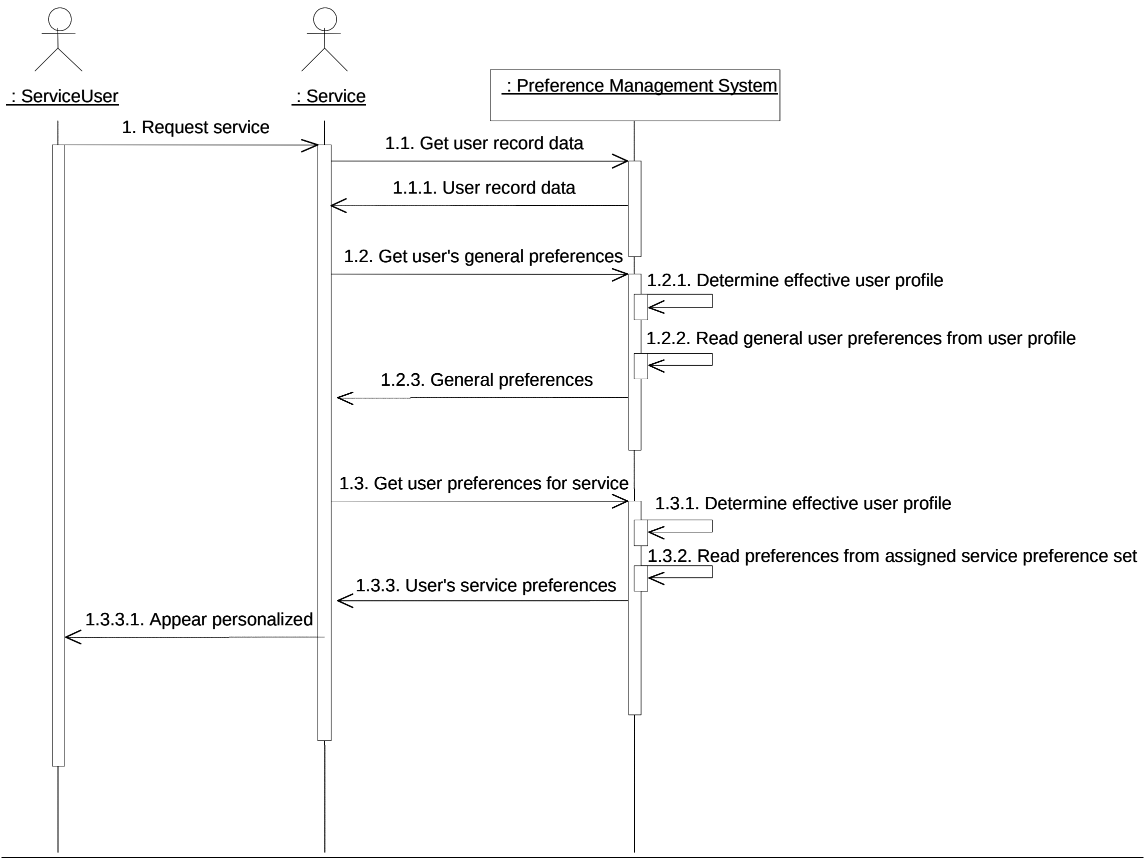

: ServiceUser
: Service
: Preference Management System
1. Request service
1.1. Get user record data
1.1.1. User record data
1.2. Get user's general preferences
1.2.1. Determine effective user profile
1.2.2. Read general user preferences from user profile
1.2.3. General preferences
1.3. Get user preferences for service
1.3.1. Determine effective user profile
1.3.2. Read preferences from assigned service preference set
1.3.3. User's service preferences
1.3.3.1. Appear personalized

Figure 3-30: Query Preferences Sequence Diagram

When the service is executing a request of the user (step 1) and can consider the user’s preferences in this execution, the corresponding preferences have to be inquired from the user directly or to be inquired from the Preferences Management System. According to the type of preference data needed, the service queries the User Record data (step 1.1), the user’s general preferences (step 1.2), or the user’s service preferences (step 1.3). The service does not have to ask for all three kinds, but can select only the appropriate category.

Obtaining the user’s general preferences as well as the user’s service preferences requires the determination of the currently effective User Profile (steps 1.2.1 and 1.3.1). Accordingly, the preference data of this User Profile are returned to the requesting service (steps 1.2.3 and 1.3.3). The retrieved preference data can now be processed and the service’s execution can be adapted to them accordingly, i.e. the services appears personalized, (step 1.3.3.1).

Manage Preferences

The users have to manage their User Profiles, i.e. to create and delete them, specify Selection Contexts for each, and to provide the general user preferences. These tasks are covered by the *manage preferences* use case. It contains all activities, which are necessary to enable the user to specify the usage of the particular service preferences. This cannot be and should not be provided by the actual services, but by the user’s home environment, which provide the user a comfortable access to these functions. For example, the user can create a User Profile for ‘home’ and a profile for ‘office’ and specifies the Selection Context for both profiles by setting suitable time switches and location relationships. Now, the user can specify the service preferences in the services directly and choose one of the created User Profiles, in which the service preferences should be used thereafter.

The data contained in the User Record are to be maintained by the home environment provider. The user cannot change these data. The data, to be managed by the user, confines to the User Profiles and the general user preferences. Accordingly, the following tasks can be derived:

- To create a User Profile (e.g. by copying of another profile)
- To delete a User Profile
- To change a User Profile(e.g. update user-given name or description)
- To activate a certain User Profile, i.e. to explicitly specify a User Profile to be effective without considering the Selection Context
- To specify the Selection Context of a User Profile
- To specify the general user preferences

These tasks can be described by according refined use cases as shown in the use case diagram in Figure 3-31.

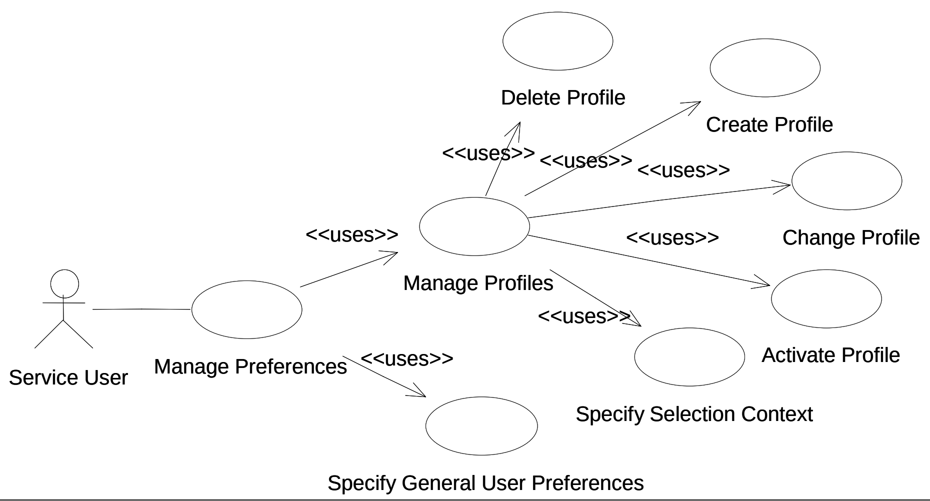


Figure 3-31: Manage Preferences Use Cases

Whereas the provisioning of the service specific preference settings is to be realized by the services themselves, the general management tasks are to be provided by an additional and service independent component. This component has to enable the users to manage their own User Profiles, depicted in Figure 3-31 as the 'Manage Preferences' and related use cases.

3.3.5 Functions of the Personalization Model

Based on the analysis of the personalization usage scenarios and relevant use cases, the complete functions, which have to be provided by the Preferences Management System, can be summarized as follows:

- Preference provisioning
    - Obtain list of user's profiles
    - Store preferences
- Preference query
    - Retrieval of user's service preferences assigned to a profile (requires determination of the currently valid User Profile)
    - Retrieval of data stored in the User Record
    - Retrieval of user's general preferences

- Preference management
    - Creation of profiles
    - Deletion of profiles
    - Changing or updating of profiles
    - Selecting of a profile
    - Setting the Selection Context of a profile
    - Specifying the general user preferences

As described in section 3.3.4.1, there different roles according the access to the personalization function can be summarized to the:

- Internal access (HE-VASP and Service Adaptation Function)
- External access (external VASP)

Accordingly, two interfaces to the personalization functions are introduced for each type of access – the *internal* and the *external* interface as depicted in Figure 3-32.

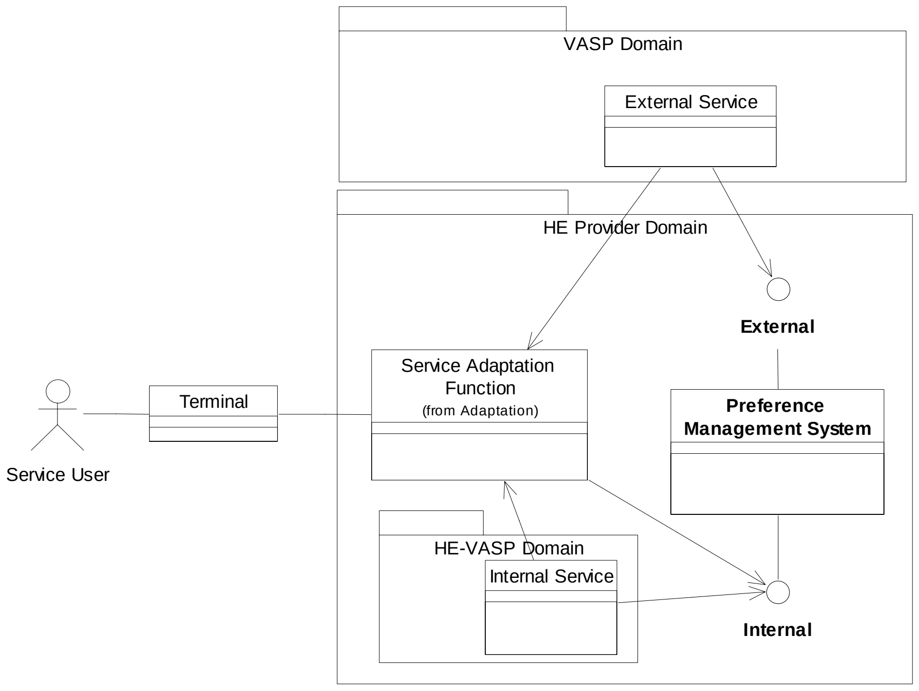


Figure 3-32: Internal and External Interfaces of the Preferences Management System

According to the discussion of the different actors and their access rights to the personalization function and preference data, the identified functions have to be assigned to these interfaces. Table 3-4 shows the functions, which have to be provided in the two interfaces respectively.

Function	***Internal Interface***	***External Interface***	***Comments / Relevant Use Cases (PP: Provide Preferences, QP: Query Preferences, MP: Manage Preferences)***
List User Profiles	✓	✓	PP, QP, MP
Store Service Preferences	✓	✓	PP
Get Service Preferences	✓	✓	PP (services can only retrieve their own preferences)

Function	***Internal Interface***	***External Interface***	***Comments / Relevant Use Cases (PP: Provide Preferences, QP: Query Preferences, MP: Manage Preferences)***
Get User Record Data	✓	✓	PP, QP, MP (not all data have to be visible)
Get General User Preferences	✓	✓	PP, QP, MP
Create User Profile	✓	✗	MP
Delete User Profile	✓	✗	MP
Update User Profile	✓	✗	MP
Select User Profile	✓	✓	QP, MP (service might also enable the user to switch the profile)
Set Selection Context	✓	✗	MP
Specify General User Preferences	✓	✗	MP

Table 3-4: Personalization Functions

The following sections introduce each function in detail. The specifications use an abstract form to specify the individual functions, i.e. no specific programming or interface definition language is used. However, the specifications contain all necessary information to derive concrete interface specifications according to the selected implementation platform.

It is furthermore assumed that the component, which calls the proposed functions and therefore denoted as the client or caller, is always identified, e.g. by a corresponding ‘ServiceIdentifier’. Similarly, the identifier of the particular user, i.e. ‘UserIdentifier’, is known in the method body. How this is realized depends on the actual implementation platform and implementation concept, e.g. by knowing the address of the client from the method call, by introducing additional parameters, e.g. as session identifier, or by providing single interfaces instances, e.g. to each client or for each user. As already mentioned above, this is very implementation dependent and is therefore not defined in this specification. Therefore, the specifications, as introduced in the following, do not reflect these implementation related issues.

Furthermore, it is assumed that according to the personalization information model (section 3.3.3) there are special data elements, such as the Default User Profile, default general user preferences set, and the default service preferences sets. These information objects are supposed to be always there, i.e. they cannot be deleted as long as the user account, i.e. the User Record, exists.

In order to remove Attributes, the following specifications do not define according delete functions. Instead, an attribute is removed if it is set to an empty value, e.g. null or ‘\0’. The implementation of a method should remove the attribute from the information object in this case. This simplifies the function and interface specification.

3.3.5.1 List User Profiles Function

The ‘list User Profile’ function can be realized by specifying an according method:

```
listUserProfiles():UserProfileList
```

This method looks up the User Record of the given user identifier to collect references to the User Profiles contained in it. It creates a list of the User Profile names, i.e. the ‘UserProfileList’, which is returned to the caller. The ‘UserProfileList’ therefore is defined as a list of User Profile identifiers, i.e. the names given by the user.

3.3.5.2 Store Service Preferences Function

This function stores the service preferences, which the user has specified during the service usage. The service preferences are described as Attributes, i.e. as name/values pairs. In addition to the service preferences, the according User Profile, in which the preferences shall be stored, is needed. The function looks up the given User Profile, selects the appropriate service preferences set, i.e. the set, which belongs to the service, and stores the given service preferences into this set. The signature of the corresponding method can be defined as follows:

```
storeServicePreferences(AttributeList, UserProfileName):resultCode
```

If there are preferences already specified in the set, they are overwritten. The other preferences of this set remain untouched.

3.3.5.3 Get Service Preferences Function

In order to retrieve the service preferences to be considered in the service provisioning the following method can be called:

```
getServicePreferences():AttributeList
```

First, the currently effective User Profile has to be determined. If there is a User Profile, which is explicitly activated by the ‘select User Profile’ function, this User Profile has to be chosen. Otherwise, the list of User Profiles has to be traversed and each Selection Context has to be evaluated. The first valid User Profile, which is the effective User Profile, is searched for the reference to the service preference set belonging to the calling service. If there is no valid User Profile, the Default User Profile is chosen instead. The attributed contained in the service preference set is returned as a list of Attributes. It is up to the actual implementation whether, if the default values for the Attributes are included in the returned list or if only the Attributes, which are changed by the user, are included in the returned list.

3.3.5.4 Get User Record Data Function

This function can be simple realized by a method as follows:

```
getUserRecordData():AttributeList
```

Because the data in the User Records are also name/value pairs, they can be returned as a list of Attributes. The specification of the presented Personalization model does not include specific attributes for the User Record. These are up to the implementation and to the requirements of the runtime system. They have therefore to be defined by the home environment provider, i.e. by the system operator.

3.3.5.5 Get General User Preferences Function

Similar to the ‘get service preferences’ function, this function defines a method as follows:

```
getGeneralUserPreferences():AttributeList
```

Likewise, this method has first to determine the effective User Profile, either the explicitly selected one, the first User Profile, whose Selection Context is valid, or the Default User Profile. Then, the Attributes of the general user preferences set belonging to the determined User Profile are returned as a list of Attributes.

3.3.5.6 Create User Profile Function

This function creates a new User Profile. Because a User Profile can contain several relevant data, which have to be specified in order to obtain a functional User Profile, it is strongly suggested to create a new User Profile only by copying and renaming of another one. When the User Record is created, i.e. at the time, when the user account is initialized the first time, all required data structures including the Default User Profile and default preferences set are created, too. When the user wants to create the first User Profile, the default profile should be copied and renamed. Therefore, the corresponding method needs only the name of the User Profile, which shall be the base for the copy, and the name of the new User Profile. It is defined as follows:

```
createUserProfile(OriginUserProfileName, NewUserProfileName):resultCode
```

This methods expects that the Default User Profile cannot be deleted, i.e. that there is always an existing User Profile to make a copy of. The copying process has furthermore to ensure that also all the preferences sets, which are referenced from the origin User Profile, are copied as well. Of course, the implementation should apply some suitable algorithms for this, i.e. copy-on-write, in order keep the system efficient.

3.3.5.7 Delete User Profile Function

The ‘delete User Profile’ function just needs the name of the User Profile, which is to be removed from the system. The according method looks as follows:

```
deleteUserProfile(UserProfileName):resultCode
```

All the data structures, i.e. the corresponding general user preferences set as well as the corresponding service preferences sets are deleted as well. It depends on the actual implementation, whether all the data are really deleted or if they are just hidden for a possible restoration later. This might be reasonable for users who did a mistake unintentionally or for situations where the system lapses. However, it has to be ensured that the default data structures including Default User Profile and default preferences sets cannot be deleted unless the complete user account, i.e. the User Record, is to be removed from the system.

3.3.5.8 Update User Profile Function

Each User Profile contains a number of Attributes. The specification of this Personalization model only defined the name and the category attribute. Nevertheless, there might be more in a real system. In order to update these Attributes, the following method is defined:

```
updateUserprofile(UserProfileName, AttributeList):resultCode
```

The method looks up the corresponding User Profile from the User Record. The Attributes of this User Profile are overwritten by those specified in the given list of Attributes, or added, if the User Profile did not have these Attributes.

3.3.5.9 Select User Profile Function

The explicitly selection of a User Profile eliminates the automatic selection by evaluation of the profiles’ Selection Contexts. As long, there is a User Profile, which is explicitly selected by this function, this User Profile has to be chosen for the provisioning of preferences data. There can be always only one User Profile to be selected explicitly. If the according method is called again, the User Profile, which was selected before, is automatically deselected. This means, in the implementation there should be an according reference to the selected profile, e.g. a hidden attribute. The corresponding method looks as follows:

```
activateUserProfile(UserProfileName):resultCode
```

Of course, the explicit selected User Profile has also to be deselected in order to activate the automatic selection process as defined by the Selection Context conditions. This is done by the following method:

```
deactivateUserProfile():resultCode
```

After this method call, the currently effective User Profile is again determined by the evaluation of the Selection Context conditions.

3.3.5.10 Set Selection Context Function

The Selection Context defines whether a User Profile is active. This means the Attributes of a Selection Context are conditions, which have to be processable and whose evaluation has to result in a Boolean value, i.e. true or false. In order to check, if a User Profile is currently valid, all Attributes of the assigned Selection Context are evaluated. If the result of all evaluations is true, the User Profile is valid, i.e. effective. If even only one condition cannot be evaluated to true, the User Profile is not valid.

The evaluation process has to know the meaning of each condition attribute, i.e. whether it is a time switch or a location area specification. This means, according to the name of the attribute the suitable evaluation algorithms have to be triggered. The Personalization model introduced in this work does not specify concrete types of conditions, because these are dependent on the actual system, i.e. available data to be used, etc. The model only proposes some examples, such as the time switches for the timely dependency of User Profiles, the location for location-based User Profiles, and the terminal for terminal-based User Profiles.

Therefore, the method to implement this function uses attributes for the specification of the Selection Context, i.e.:

```
setSelectionContext(UserProfileName, AttributeList):resultCode
```

This method searches the corresponding User Record to obtain the reference to the target User Profile with the given name. The attributes from the given list are copied into the Selection Context assigned to this profile. In order to obtain the Selection Context of a User Profile the following method can be used:

```
getSelectionContext(UserProfileName):AttributeList
```

This returns all condition attributes of the Selection Context assigned to the User Profile with the given name.

3.3.5.11 Set General Preferences Function

In order to set the general preferences belonging to a User Profile the following method can be called:

```
setGeneralPreferences(UserProfileName, AttributeList):resultCode
```

The User Record is searched to obtain access to the target User Profile with the given name. Then, the Attributes in the given list are copied into the general preferences set assigned to this User Profile. The Attributes, which are already stored in this set, are overwritten; the others are added.

3.3.6 Personalization Interface Specification

As described in section 3.3.5, the access to the personalization functions and hence to the personalization data is to be restricted according to the role of the client. The different access schemas are classified to the *external* and the *internal* access. Accordingly, two types of interfaces can be defined:

- The external interface, and
- The internal interface

The component, which implements and provides the identified personalization functions, is described as the *Preferences Management System* in the following. It implements both interfaces and has to secure the access to them. This means, the clients have to be identified and the access to the different functions and data has to be granted accordingly.

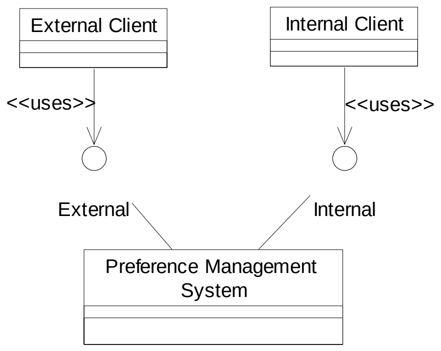


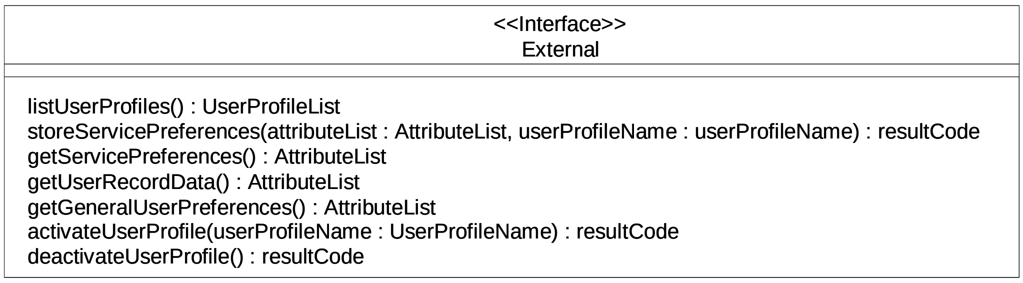
Figure 3-33: Usage of Personalization Interfaces

The Preferences Management System provides both interfaces, which are to be used by the clients. It is up to the implementation and the selected platform how the interfaces are realized internally. For example, there could be a core class implementing all methods and additional classes for each interface, which forwards the method calls to the core class object according to the access rights definition. The following two sections introduce a more detailed specification of both interfaces.

3.3.6.1 External Interface

The external interface specifies methods to implement the functions, which are associated with the external access as described in Table 3-4 in section 3.3.5. Figure 3-34 shows the specification as an UML[21] diagram.

[21] UML: Unified Modeling Language [UML1, UML2, HENNICKER]



<<Interface>>
External
listUserProfiles() : UserProfileList
storeServicePreferences(attributeList : AttributeList, userProfileName : userProfileName) : resultCode
getServicePreferences() : AttributeList
getUserRecordData() : AttributeList
getGeneralUserPreferences() : AttributeList
activateUserProfile(userProfileName : UserProfileName) : resultCode
deactivateUserProfile() : resultCode

Figure 3-34: Specification of the external Personalization Interface

The basic data types, such as the 'AttributeList' and the 'UserProfileName', are not defined in this specification. They have to be defined by the implementation, because their formats depend on the selected target platform and on the realization of the individual data elements, i.e. the 'UserProfileName' could be a string or an index of a corresponding database set.

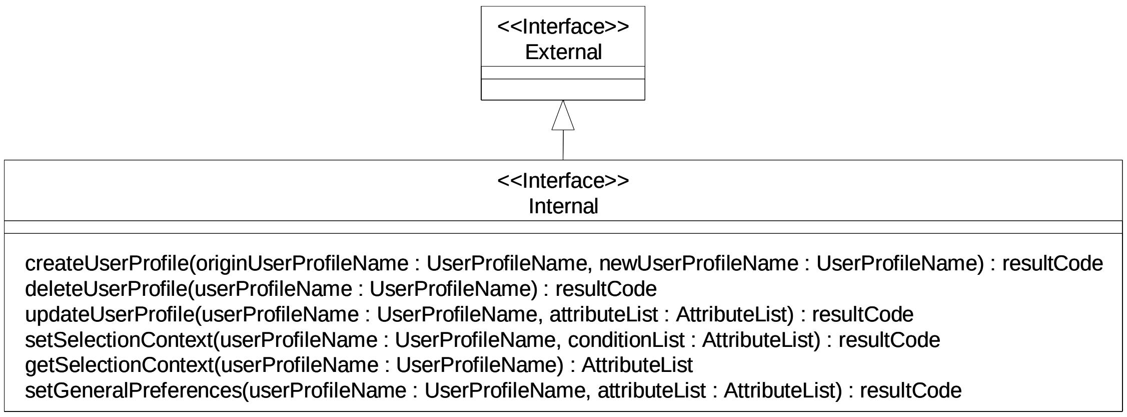
This means, from this given specification, concrete interface specifications in the corresponding interface definition or programming language have to be derived. An exemplary Java interface looks as follows:

```
package Personalization;
public interface External {
   public UserProfileList listUserProfiles();
   public resultCode storeServicePreferences(AttributeList attributeList,
userProfileName userProfileName);
   public AttributeList getServicePreferences();
   public AttributeList getUserRecordData();
   public AttributeList getGeneralUserPreferences();
   public resultCode activateUserProfile(UserProfileName
userProfileName);
   public resultCode deactivateUserProfile();
}
```

This example depicts a preliminary interface to be used in the implementation. The necessary data types have still to be defined.

3.3.6.2 Internal Interface

The internal interface specifies methods to implement the functions, which are associated with the internal access as described in Table 3-4 in section 3.3.5. Figure 3-35 shows the specification as an UML diagram.



<<Interface>>
External
<<Interface>>
Internal
createUserProfile(originUserProfileName : UserProfileName, newUserProfileName : UserProfileName) : resultCode
deleteUserProfile(userProfileName : UserProfileName) : resultCode
updateUserProfile(userProfileName : UserProfileName, attributeList : AttributeList) : resultCode
setSelectionContext(userProfileName : UserProfileName, conditionList : AttributeList) : resultCode
getSelectionContext(userProfileName : UserProfileName) : AttributeList
setGeneralPreferences(userProfileName : UserProfileName, attributeList : AttributeList) : resultCode

Figure 3-35: Specification of the internal Personalization Interface

The internal interface specifies the function, which have restricted access. It inherits additionally the function from the external interface. This means, from this given specification, concrete interface specifications in the corresponding interface definition language or programming language have to be derived. An exemplary Java interface looks as follows:

```
package Personalization;
public interface Internal extends External {
   public resultCode createUserProfile(UserProfileName
originUserProfileName, UserProfileName newUserProfileName);
   public resultCode deleteUserProfile(UserProfileName userProfileName);
   public resultCode updateUserProfile(UserProfileName userProfileName,
AttributeList attributeList);
   public resultCode setSelectionContext(UserProfileName userProfileName,
AttributeList conditionList);
   public AttributeList getSelectionContext(UserProfileName
userProfileName);
   public resultCode setGeneralPreferences(UserProfileName
userProfileName, AttributeList attributeList);
}
```

The interface inherits from the external interface, which means that the external functions, i.e. the methods provided in the external interfaces, are also available in the internal interface.

3.3.7 Functional Decomposition of the Personalization Model

This section describes a proposal for the functional decomposition of the Preferences Management System. There are different possibilities to implement the proposed Personalization model. However, the following proposal depicts a general guideline introducing required functional elements.

According to the specification of the Personalization model, four different functional elements can be identified: the preferences manager, the profile storage, the preferences storages, and the access control. Each has its own distinct functionality and is clearly separated from the others. Figure 3-36 shows an overview of the functional separation.

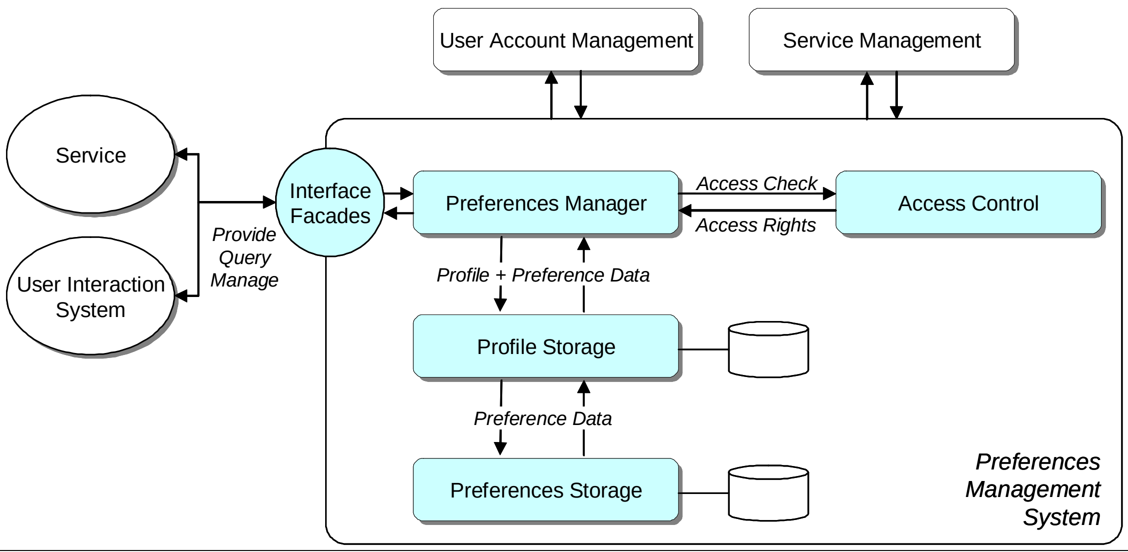


Figure 3-36: Functional Separation of the Preferences Management System

The complete Preferences Management System provides the identified personalization interfaces (introduced in section 3.3.6). These interfaces, depicted as the interface facades, are actually implemented by the preferences manager. This means, the method calls are executed in this component. The Preferences Manager accesses the preference data by using the profile storage, which in turn uses the preferences storage elements. It has to check the access right of the clients by interrogating the access control component. This component maintains an access control list, which specifies the individual access rights for each type of client, i.e. external or internal service, or administrative access.

Reasonable design patterns [GAMMA] should be selected for the realization of the identified functional elements. Figure 3-37 shows an exemplary approach how design patterns can be used in order to obtain an effective and flexible implementation specification.[22]

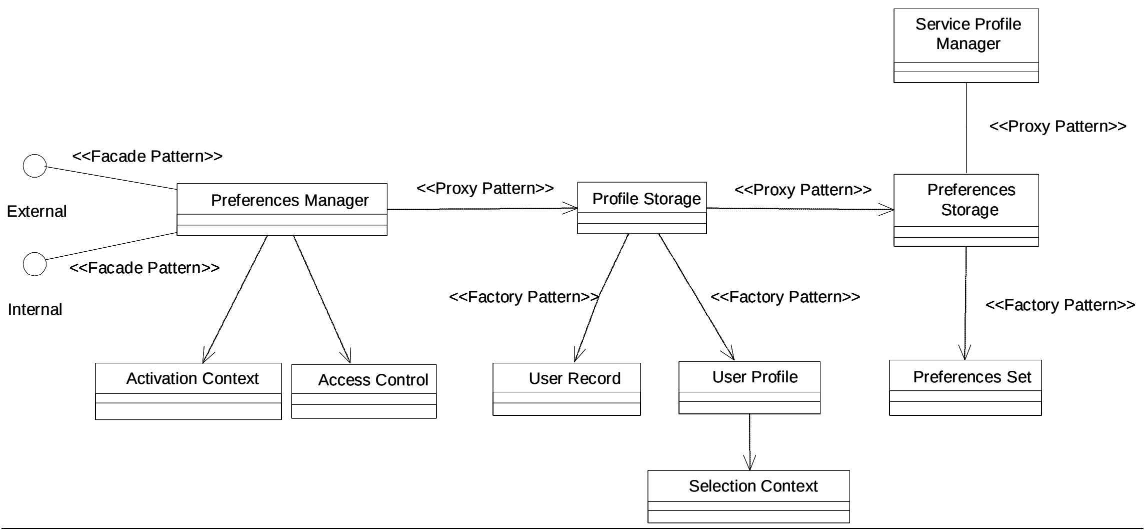


Figure 3-37: Applied Design Patterns in the Personalization System

The following sections introduce each of the four identified functional elements more detailed.

3.3.7.1 Preferences Manager

The Preferences Manager utilizes the facade design pattern to provide a suitable interface, which hides the complexity of the internal processes. This simplifies the usage of the internal components. The Preferences Manager uses the profile storage and the access control elements in order to execute the method calls.

In addition, the Preferences Manager realizes the implicit profile selection and activation process. To decide, which profile is active, the Preferences Manager depends on an activation context object. In an I-centric User Interaction system, the activation context is related to the Delivery Context. It provides the necessary data to evaluate to conditions of the Selection Contexts of the individual User Profiles.

A User Profile becomes effective, i.e. is dynamically selected, if its associated Selection Context matches the actual activation context. The Preferences Manager has to check all User Profiles and evaluate their Selection Contexts. If there is more than one User Profile of a user with a valid Selection Context, the profile with the highest priority is chosen. If the user did not have specified priorities, i.e. all User Profiles have the same default priority, the first matching User Profile is used. If there is no User Profile with a valid Selection Context, the Default User Profile is selected. The selection and activation process is not executed, if there is a User Profile that is manually activated by the ‘select User Profile’ function.

The activation context element depicts a data structure, which provides the necessary data for the evaluation process. The format of this element should be defined according to the implementation of the applied evaluation functions. The Preferences Manager has to query the required data from the corresponding external system elements, which do not belong to the pro-

[22] See [GAMMA] for detailed descriptions of the applied design patterns.

posed Personalization model, such as the Ambient Information Server as introduced by the Ambient Awareness model (section 3.4).

3.3.7.2 Profile Storage

The profile storage element utilizes the proxy design pattern and maintains the User Record and User Profile elements by applying the factory design pattern. In this way, this element provides a high-level interface providing the functions necessary to manage User Records and User Profiles. It can be implemented as a proxy to a persistent database, which stores the actual User Record sets and User Profile data sets. These data elements can be realized practically using the factory design pattern, i.e. there is a factory, which manages all the respective instances.

Furthermore, the profile storage element uses the proxy preferences storage element in order to obtain the preferences data. Because of this separation, the profile storage element does not have to deal with the management of the actual preferences data.

3.3.7.3 Preferences Storage

The preferences storage element manages the different preferences sets of the individual users. Therefore, this element has to have access to the service profiles, which could be provided by corresponding service profile manager. This is necessary in order to provide the default values of the Attributes, if they are not specified in a preference set. The preferences sets are supposed to be stored persistently in a database. It depends on the actual implementation whether the general preferences set and the service preferences set are realized using the same or different class specifications. For the management of the preferences sets, the preference storage element uses the factory design pattern, which means that there is a factory object, which takes care about the instances of these preferences set elements.

3.3.7.4 Access Control

The access control element maintains access control list, which defines the access rights for the different clients. On the hand, the access rights depend on the type of client, i.e. internal or external component, but on the other hand also on the user, who has initiated the request. Naturally, users are only allowed to see and to change their own preferences. The complete access control is allocated to this functional component to simplify the implementation of the other components and to have a single source for the specification of the access rights. This means, the access control component provides the necessary functions in order to authenticate and authorize the requests from the clients. In this way, the other functional components of the Preferences Management System need only to ask the access control whether a certain request is allowed or not. If they receive a positive result, they continue with the execution of the request, if not they have to respond to the client accordingly.

The prerequisite for the access evaluation is the identification of the clients. It does not necessarily mean to identify each client, but only to determine the group to which the client belongs, i.e. external or internal. This can be achieved by maintaining databases, which manages the service deployment data including (network) addresses and access role. Furthermore, commonly available security concepts could be applied, such as the public-key cryptography mechanisms, e.g. the Public-Key Cryptography Standards (PKCS) developed by RSA Laboratories. These include digital signature mechanisms, which can be used to authenticate a client and also to ensure the identity of the sender as well as the integrity of the request.

However, simple usage of digital signatures may not be enough. Without a proper design, simple eavesdropping attacks can easily use intercepted requests and corresponding responses to feint the access control system. Therefore, encryption methods should be applied in the communication with external components. In an actual implementation, this can be realized in the network layer, i.e. to use trustworthy network infrastructures like IPSec, or in the transport layer, i.e. suitable secure protocols like HTTPS should be chosen for the communication.

Finally, it hast to be remarked again that the ensuring of the privacy of the user should the highest commandment for the development of the personalization system. Therefore, suitable design decisions and carefully selections of security concepts have to be enforced in the development process. Section 3.3.8 discusses some further security issues, which have to be considered in the development as well as in the operation of the personalization system.

3.3.8 Discussion of Security and Privacy Concerns

Security and privacy concerns are important aspects of any personalization concept. Today, there is no common standard, which covers all security and privacy issues for protecting personalization information as used in communication systems. The following sections give a brief overview of potential security threats related to the management of personalization data as well as available mechanisms to counter these threats.

During the specification process as well as before the start of the commercial operation of a system dealing with personal data, all possible security issues, which possibly can threaten the user's privacy but also the system stability, have to identified and according countermeasures have to be implemented. Generally, the requirements, which can be deduced from the analysis of the identified security threats, can be summarized as follows:

- The access rights have to be defined carefully
- Reliable identification mechanisms have to be executed before granting access
- Only authorized access according to specified access rights has to be ensured
- Secure communication ensuring the integrity of transferred data has to be applied

In order support these requirements, suitable network, operating system, and software platforms have to be selected for the implementation of the personalization system and of the corresponding interfaces, which are provided to external components. There are a number of available and easy-to-integrate security solutions, like Secure Socket Layer (SSL) with strong encryption and X.509 certificate-based authentication.

[BARBIR] identifies seven security threats, whereas three of them are relevant to the personalization approach as presented in this thesis. The following sections discuss these threats in detail.

3.3.8.1 Malicious Entity accesses User Data

This security threat endangers the privacy issue concerning the user's personalization data stored in the personalization system. A malicious entity could be capable to access user personalization data directly, i.e. it can read the information stored in a User Record, a User Profile, a Selection Context, a general user preferences set, or a service preferences set of a certain user. This would compromise the privacy of the user.

To prevent this, proper authorization before accessing any information stored within the personalization system is essential. The home environment provider and an external value added service providers differ in their granted privileges, i.e. they have different access rights to certain preference management functions. This ensures that the personalization system protects the stored data ensuring the privacy of the user. Because of their external role, external services cannot be given the same trust status as the home environment services. Without this restriction, malicious external components could easily access the information stored by other service. For example, such a malicious entity could analyze the usage behavior of other services and misuse the personal information about the users. This would mean a huge violation of the privacy and the users would loose the trust in the system and would stop using it. The proper authorization of the accessing component as well as the correct and complete enforcements of the defined access limitation is an inalienable prerequisite to avoid this security thread. To realize this, suitable software technologies have to be selected for the implementation supporting the necessary security mechanisms, e.g. authorized and secure interface access.

3.3.8.2 Intercepting User Personalization Data during Transmission

This threat concerns the unauthorized access to personalization data during the transmission between the personalization system and the inquiring service. A malicious and unauthorized entity could be capable to intercept user personalization data during transmission, i.e. it can read the transmitted user's preferences and can draw conclusions about the created User Profiles, about subscribed services, as well as the specified Selection Context. The impact of this security threat is similar to the threat described above, although the necessary solutions to prevent this are different. Here, the data itself and not only the access to it have to be protected.

To avoid this unauthorized access, the transfer of sensible information has to take place in a secure manner, i.e. the personalization system has to assure that the transferred data cannot be read by unauthorized entities. This can be realized by applying suitable encryption technologies or by transmitting the data in anonymous form, i.e. without any relationship to the user. The intercepting of anonymous data preserves the privacy of the user, but it still provides possibilities to draw conclusions about the system usage.

3.3.8.3 Malicious Entity modifies User Personalization Data

In contrast to the intercepting or illegal accessing, the modification of user data depicts interferences to the system. A malicious entity could be capable to modify or delete user personalization data, i.e. it can modify or even delete the information stored in a User Record, a profile, a Selection Context, the general user preferences set, or service preferences set of a certain user. The correct functioning of the personalization system would be obstructed.

The users might not be able to use the services as usual. In this way, the fundamental principle of personalization and the accompanied advantages for the user are abrogated completely. Such incidents lead to unexpected preferences or user information of the affected user and thus to an unpredicted or unwanted service behavior. In addition to that, users and service providers have to worry about the loss of the data necessary for the successful service provisioning.

To prevent such malignant access to the personalization data, the proper authorization as well as the correct enforcements of access restrictions is indispensable. Moreover, it has to be ensured that the requests originally issued by authorized entities cannot be altered in any way before they are executed in the personalization function, i.e. the integrity of such request also needs to be preserved.

3.3.9 Conclusion

The presented Personalization model specifies an information model to describe and to structure the users' preference data. The individual preference settings can be categorized as general as well as service specific preferences and stored in different User Profiles. Each preference setting, i.e. Attribute, is modeled as a name value pair, whose type is defined by the corresponding service. The user can define a Selection Context for each User Profile, which is used to determine the effective User Profile.

According to the relevant usage scenarios and derived use cases, the Personalization model identifies necessary functions, which has lead to the specification of the corresponding interfaces for the internal and for the external access to the Preferences Management System. Furthermore, this section has identified some requirements with respect to security and privacy concerns, which have to be considered by actual implementations of this Personalization model.

3.4 Ambient Awareness Model

The Ambient Awareness model, presented in the following sections, was developed based on the identified requirements in order to support I-centric User Interaction. After the general introduction, which repeats and refines the individual objectives, the basic approach is introduced. This approach contains a mechanism to describe ambient information and to provide them to an according store, where this information can be interpreted, revaluated, and finally provided to the services and the Service Adaptation Function.

3.4.1 Introduction

The high-level objective of this model is to support an I-centric User Interaction system with an extensible and sustainable ambient aware decision support and planning system. This system provides only relevant information and/or services to the mobile user, where the relevance depends on the user's task. Ultimately, the challenge is to enable mobile users to interact intelligently with objects and services in changing environments. This can be achieved through the creation of personal devices, which utilize combinations of personal-area, local-area, and wide-area connectivity to accumulate dynamic models of environments and users. Such models enable the automatic extension and proper support of the user interaction with objects and services. It is essential that context is expressed in a human understandable form, yet still be efficiently processed by computer systems: therefore, one must understand what context is and how it can be used, and there has to be an architectural support for this.

The level of ambition was to go beyond the state-of-the-art utilizing location as an indicator of context and to work on an extensible and sustainable ambient aware decision support and planning system based on observable multi-context indicators such as:

- Physical contextual indicators, e.g. location, levels of lighting, noise, humidity, and temperature
- Computing context indicators, e.g. network connectivity characteristics, nearby devices
- User context indicators, e.g. user's location, social conditions, user's history
- Temporal context, e.g. time of day, day of week, time before or after a specific event

The rationale for the selected approach was to advance beyond today's location-based systems to achieve awareness of context that cannot be inferred from location. The availability of diverse sensor technologies is one prerequisite for a functioning Ambient Awareness model.

Beside the direct sensed ambient information, there is also indirect ambient information, which can be obtained by drawing conclusions from the direct sensed information. This requires a reasonable evaluation and interpretation of ambient information and additional facts by suitable derivation schemas. The dimension of information, which is to be considered here, is theoretically unlimited. The more information is available and the more intelligent the derivation rules are, the quality of the ambient awareness can be better. However, real systems will only consider a specific set of relevant information according to the application domain.

The interpretation and derivation of ambient information is related to the area of knowledge-based and rule-based systems. This indirect perception of context boils down to an assisted robust and reliable perception and activation of contextual mechanisms irrespective of incomplete information of knowledge about the users in their environments.

According to the general I-centric User Interaction model (introduced in section 3.1), the Ambient Awareness model has to provide a reasonable description of the respective context, in which a user interaction takes place, to the Service Adaptation Function, to the Personalization model, and in particular to the service. Beside the description of ambient information, the Ambient Awareness model has to support a number of basic functions for the management of ambient information. These include:

- Discovering and sensing
- Evaluating and interpreting
- Provisioning

The following sections describe detailed the developed Ambient Awareness model and introduce the corresponding concepts, which fulfill the identified objectives.

3.4.2 Overview

According to the vision of I-centric Communications, the user always acts in a *context*. The context describes relationships to objects in the user's communication space including physical devices, other humans, but also relevant information. *Ambient information* is a substantial part of the context description and can be defined as follows:

Ambient information describes the environment of the user at a certain moment in time. This includes physical measurable data, such as temperature, humidity, loudness, etc., but also relevant items residing in the environment, such as devices and other objects.

Accordingly, sensor networks play a major role in providing ambient information. Sensor technologies will be embedded in the mobile equipment, communication networks, living and working environments to sense the identity, the location, and the current activities of the user as well as the environmental conditions. Advances in sensor technologies are conditions to reach more fine-grained ambient information and by this an improved ambient aware behavior.

Many devices already adapt today to some degree to their operating environment. One example is the television set, which adjusts its image contrast to the ambient lighting level. 'Intelligent I/O behavior' can be used to adapt the output characteristics depending on the situational context, but also the input characteristics can be adapted in many situations. Ambient aware behavior improves the user interaction by considering additional information beside the information, which is directly communicated by the user. The users can communicate more efficiently with the system, because ambient information can complete missing inputs. For example, the user might say: 'Turn off the light!' According to the current location, the relevant sources of light can be determined and can be controlled. The user does not necessarily have to identify the individual lights. In this way, the user is not bothered with long lasting interaction. On the contrary, the users experience a pleasant and intelligent system behavior.

Obviously, the *location* extension plays a central role when describing the user's environment. On the one hand, ambient information can be obtained directly from the user and the personal devices, i.e. mobile phone, measuring instruments such as heart rate monitor. On the other hand, sensor devices, which are in the vicinity, can be considered supplementary. For example, based on the available location information, the system can look up, whether there are additional sensor devices providing relevant information about this location area. In this way, the ambient information can be gathered from different sources and the location can be regarded as a very basic ingredient of the ambient information.

However, the location reference does not fundamentally play an important role in the interpretation. Nevertheless, there has to be a certain relationship to the user, which can be described in different ways. For instance, the user's terminal may provide ambient information such as temperature. Even if no location data is available, this information should be provided as the available ambient information with the corresponding relationship to user. This means, although ambient information has usually a certain location relationship, the ambient aware system must also provide non-location related data with the according reference to the user.

Because of the variety of information, the structuring and the management of ambient information is a complex challenge. On the one hand, it is difficult to decide which data is relevant and, on the other hand, is not trivial to express such data suitable for further processing. Today, there are numerous approaches in several research projects following different concepts. Usually, these approaches are suited to the particular pursued application scenarios and are not universally applicable.

To which extend ambient information is to be collected depends on the requirements derived from the application fields and on the available sensor technologies. Accordingly, a flexible and extensible approach to manage ambient information is needed.

Ambient aware behavior is reached by considering available and relevant ambient information during the service execution. Accordingly, ambient awareness can be defined as follows:

Ambient awareness means to consider and to adapt to information about the user's environment (ambient information) in the service provisioning.

This means, an ambient aware service adapts to the current environment of the user. This regards to the service execution including the interaction with the user. For example, according to the user's current activity, e.g. derived from the user's heart rate sensed by a heart rate monitor and from the user's mobility, an ambient aware music player can select most suitable songs for the particular situation. The volume is automatically adjusted according to the environmental loudness to provide a most comfortable experience. If the user's situation changes, e.g. the user meets another person and starts a conversation, the music player adapts accordingly by interrupting the music or by reducing the volume.

In order to realize ambient awareness, first of all ambient information have to be gathered. Basically, ambient information can be obtained in two ways from different kinds of sources:

- Directly:
    - From sensor devices, which measure and provide physical values
    - From explicit user statements, by which the user himself describes the environment, e.g. specifying the current location
- Indirectly:
    - By interpreting available information to obtain revaluated information, e.g. if the location of a terminal, the user currently uses to interact with the system, is known, the location of the user can be derived accordingly.

To realize the gathering of ambient information from explicit user statements as well in the indirect way, the according components, i.e. the Service Adaptation Function or special interpreters respectively, have to interpret the raw data and to provide them in a suitable format to the ambient aware elements, i.e. the Service Adaptation Function again and the services.

3.4.2.1 Gathering Ambient Information from Sensor Networks

Sensor networks, as depicted in Figure 3-38, provide raw data, which can be interpreted and revaluated to obtain reasonable ambient information.

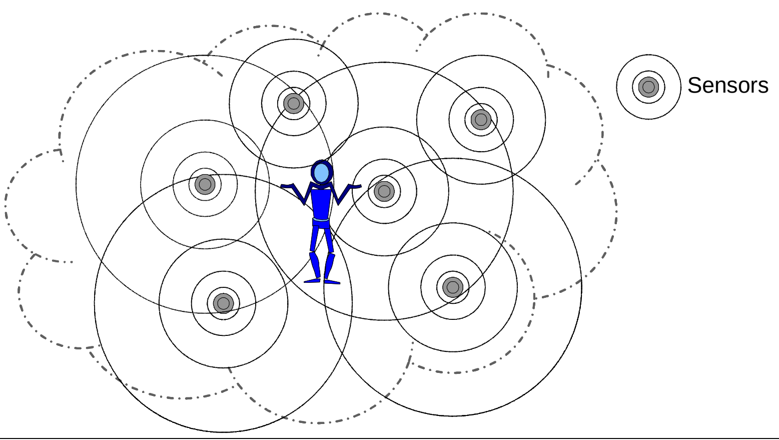


Figure 3-38: Network of Sensors gathering Ambient Information

Today, there is a variety of sensor technologies, and there is still an ongoing development to increase the quality and to downsize the magnitude of the sensing devices. The range of application domains is huge including the automotive sector, industrial factory constructions, and increasingly home/office installations. To develop improved sensor technologies and according concepts to build suitable sensor networks is still subject of international research activities. The economic and technical expectations seem to be promising. Beside the actual measurement and conversion into digital presentation, the main research focus is on the delivery and management of sensor data. Because of the amount of data and frequency, this is the very crucial point of efficient sensor networks.

Figure 3-39 depicts the possible ways to obtain ambient information.

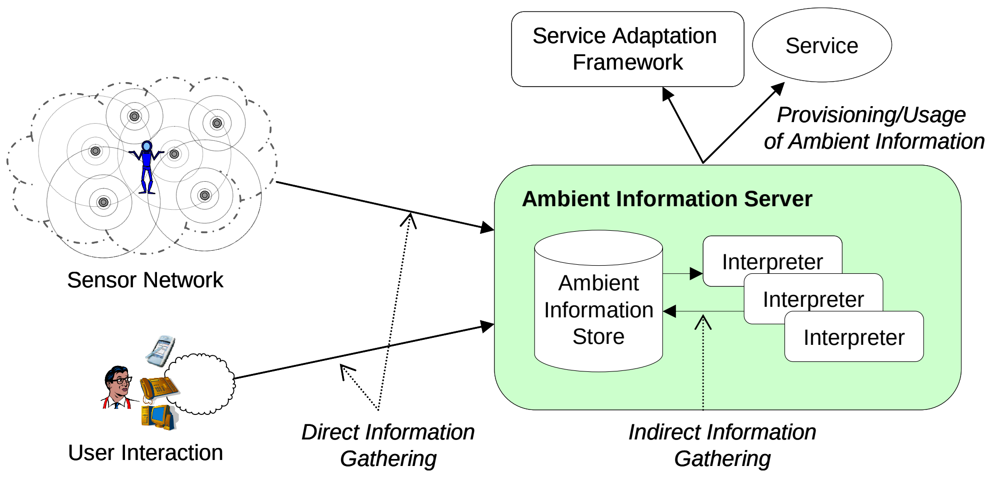

Service Adaptation Framework
Service
Provisioning/Usage of Ambient Information
Ambient Information Server
Ambient Information Store
Interpreter
Interpreter
Interpreter
Sensor Network
User Interaction
Direct Information Gathering
Indirect Information Gathering

Figure 3-39: Gathering of Ambient Information

3.4.2.2 Deriving Ambient Information from User Interaction

As the I-centric Communications concept orientates on the human, the I-centric User Interaction concept tries to improve the user interaction by following and imitating the capabilities of human-human interaction. Human-human interaction does not only consider *what* is communicated, i.e. the content, but also *how* it is communicated, i.e. the presentation style. It is a very fundamental characteristic to consider the behavior of the interlocutor. This could include for example response time or tone and style of the input, e.g. short and concise, gabby, nerved, disturbed, etc.

For human beings it is very natural to derive necessary information from the presentation style and to react accordingly. Today, such behavior is nearly impossible for computer-based artificial systems. However, the different user interface technologies are continuously improved to approach human quality in the future. Existing speech-based systems can already recognize additional characteristics of the spoken language, such as volume, emphasize, or prosody. Based on such data, contextual information can be derived by according rule-based systems. In this way, also environmental conditions could be determined, as for example the loudness in the surrounding. According to its definition, such information characterizing the user's environment belongs to the ambient information, likewise.

It depends on the specific user interface technology and terminal, what kinds of ambient information can be captured from the user interaction. One the one hand, the terminal might provide ambient information, if it is equipped with corresponding sensor devices. For example, there are already first mobile phone with temperature and location sensors. These data can be provided as ambient information, and the interaction can be adapted accordingly. On the other hand, from the behavior of the user, certain conclusions can be drawn some times. If for example the user accesses the system from a fixed terminal, such as the desktop PC in the office, it can be assumed that the user resides on the same location, i.e. in the office. This kind of drawing conclusions is quite simple, but do not require sensor networks installed everywhere. It just requires a certain logic in the system, which gathers and manages the ambient information.

If technologies capable to derive ambient information from the user interaction are employed in the realization of the user interfaces, the corresponding components have to be regarded as a source of ambient information, likewise. Accordingly, they should store the information to the Ambient Information Server and in this way provide this information to the entire system.

In principle, this way of obtaining ambient information depends on the qualities and capabilities of available user interaction technologies and will play the more important roles in future systems when suitable technologies will be available.

3.4.2.3 Revaluating Ambient Information

First, sensors provide basic information. This information can be revaluated by according *interpreters*, which understand the meaning of the information and which are able to derive enriched information from it. This is designated as *indirect information gathering*. For the revaluation, suitable rules have to be defined and to be executed by the interpreters. Whenever, new ambient information (*I*) is available and there is an interpreter capable to process this data and implementing a reasonable algorithm (*A*), this interpreter is activated and the resulting data (*I'*) is again provided to the ambient information store, i.e. I' = A(I). This process can be carried out as long as there are further interpreters, which in turn can derive extended information from the enriched information base, as depicted in Figure 3-40.

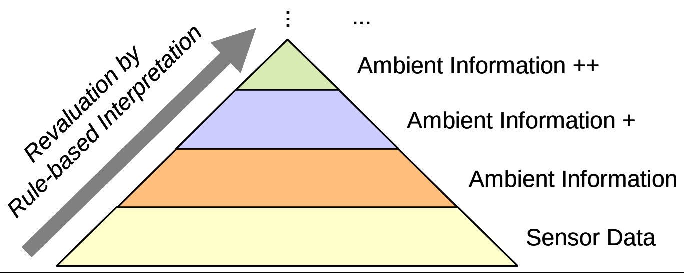


Figure 3-40: Revaluation of Ambient Information

With each interpretation loop, the resulting information is semantically enriched, but the accuracy is deteriorated, likewise. When using ambient information the sources as well as the degree of correctness and unambiguousness have to be taken into account in order to achieve a reasonable behavior. In addition, the more complex the information bases and the applied interpretation rules are, the more difficult it is for the user to understand and hence to accept the behavior of such ambient aware systems. Such concepts should be applied in real-world systems with carefulness leaving the full control of the system to the user, i.e. to decide about the interpretation rules, and watching the reaction of the user.

3.4.3 Ambient Information Data Model

Because of the manifold of different kinds of ambient data, a suitable and flexible model to structure ambient information is needed. The following sections introduce an XML-based language, which can be used to exchange ambient information. This approach is not limited to specific kinds of ambient data and is not limited to specific technologies for the sensing and provisioning of ambient information.

3.4.3.1 AmbientProfile

Ambient information is always associated to a location area and has always a temporary reference. This has to be considered in the description of ambient information. A certain set of ambient information is described by an *AmbientProfile* element, which is characterized by the location area and a number of attributes (ambient data elements), which contain the specific ambient information data.

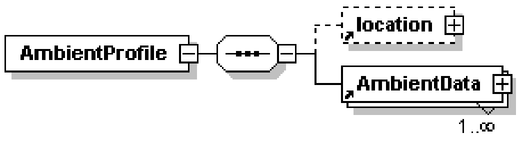


Figure 3-41: Ambient Profile Data Structure

Accordingly, an example of an ambient profile can look as follows:

```
<AmbientProfile>
   <location>...</location>
   <AmbientData>...</AmbientData>
      ...
   <AmbientData>...</AmbientData>
</AmbientProfile>
```

3.4.3.2 Location

The location area is described by the longitude and latitude values as well as an uncertainty shape as derived from [PRLY21MOB]. Figure 3-42 depicts the derived XML schema specification used for this data model.

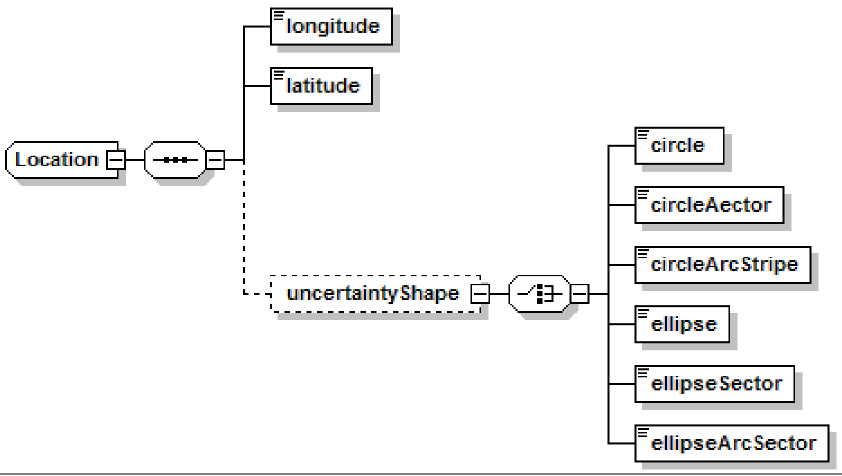


Figure 3-42: The Location Structure

The reference system, which was chosen for the coding of locations, is the World Geodetic System 1984 [WWWWGS84]. The horizontal location is defined by an ‘ellipsoid point with uncertainty shape’. The '*uncertaintyShape*' describes this ellipsoid and longitude/latitude defines the position of the uncertainty shape. According to this specification, a location area can be described as follows:

```
<location>
   <longitude>13.18831E</longitude>
   <latitude>52.31555N</latitude>
   <uncertaintyshape>
      <circle unit=”meter”>300</circle>
   </uncertaintyshape>
</location>
```

According to the application domains and to the respective location types, e.g. out-door and in-door, used in these domains, there are different approaches to identify and to describe location areas. For example, postal addresses, zip codes, and room identifiers can be used. These are much more convenient to the human being, because people use them in their daily life.

Nevertheless, a common base system is needed to manage location data. The geodetic system was chosen, because is depicts the most flexible approach to describe location areas and is applied by the considerable positioning technologies such as the Global Positioning Service (GPS) [GPS]. It can be used to describe locations with an accuracy[23], which suitable for most application scenarios, and is therefore appropriate for all kinds of location-based systems. Different positioning methods can be mapped to the geodetic system easily. For example, an active badge sensor is usually installed in a room and is therefore assigned to a room identifier, e.g. room ‘FR5101a’. The mapping rules should allow transforming this identification to geodetic data representations, which then can be used also by services, which are not aware about the specific room schema.

Beside the interaction facet, the mapping rules are an important aspect of the user interaction, too. The user must not be burdened with the technical presentation of location data, but with human understandable way, at best in the personal coordinate system, i.e. using location identifiers, which the individual knows. This means, whenever location data are to exchanged in the interaction with the user, they should be first translated into the user’s favorite presentation.

3.4.3.3 Ambient Data

The actual data describing ambient information are stored in the '*AmbientData*' segment. Each is described with a *time* stamp and a *duration* attribute, which indicate the moment when the data was sensed and the duration, how long the value is valid. The specification does not contain definitions for concrete sensor data. Instead, it uses the *any* definition, which enables to include all kinds of XML data.

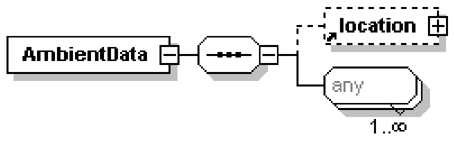


Figure 3-43: The Ambient Data Structure

[23] The accuracy of commercial GPS systems is starting from 100 meter to 10 meter. The military option of GPS as well as specific extensions, such as Differential GPS (DGPS), allows a higher accuracy up to 1 centimeter. [WWWGPS]

As depicted in the following example, the specification of the sensor data has to be provided by corresponding schema assignments (‘weather.xsd’). This schema defines the concrete data types and possible values.

```
<AmbientProfile  xmlns=”ai”
                 xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
                 xsi:schemaLocation=”ai ambient.xsd weather weather.xsd”>
   <AmbientData time=”2003-06-24T09:30:47+01:00”
                 duration=”P0Y0M0DT01H30M”>
      <location>...</location>
      <weather:temperature unit=”celcius”>28.5</weather:temperature>
      <weather:humidity>81</weather:humidity>
   </AmbientData>
</AmbientProfile>
```

If the concrete schema specification is known, the data contained in the ambient data structure can be evaluated. If the specification is not known, they have to be ignored, but the complete ambient profile is still valid.

3.4.3.4 Sources

Beside the actual value, it is important to know the source, which has provided the ambient information. Basically, the information can be provided by:

- Sensor, which directly has measured the value, or
- An interpreter, which has evaluated other available ambient information to derive extended ambient information, which is then also part of the ambient profile

The later depicts the indirect information gathering as described in section 3.4.2. The ambient data, therefore, has to be described with a *source* attribute as depicted in following example:

```
<AmbientData time=”2003-06-24T09:30:47+01:00”
             duration=”P0Y0M0DT01H30M” source=”TEMP456”>
   <location>...</location>
   <weather:temperature unit=”celcius”>28.5</weather:temperature>
</AmbientData>
```

The source attribute is optional and can contain an identifier of or a reference to the source object. The used identifiers have to be unique and have to be managed by the runtime system. In this way, the system can be interrogated to obtain more information about the source.

3.4.3.5 Quality

According to the source, ambient information has a certain creditability. Whereas information provided by sensors can be rated as being reliable, the information provided by interpreters should be rated as being less reliable. This is reflected in the data model by the *quality* attribute, with a value ranging from ‘0.0’ to ‘1.0’ as depicted in the following example:

```
<AmbientData time=”2003-06-24T09:30:47+01:00”
             duration=”P0Y0M0DT01H30M” source=”TEMP456” quality=”0.9”>
   <location>...</location>
   <weather:temperature unit=”celcius”>28.5</weather:temperature>
 </AmbientData>
```

The quality value ‘1.0’ describes a reliable value and the quality value ‘0.0’ describes an unreliable value.

3.4.3.6 References of revaluated Data to original Data

As described in section 3.4.2 by the *indirect information gathering* procedure, ambient information interpreters generate new information based on available ambient information. The new generated information extends the ambient profile with a new ambient data record. It is reasonable to know the original data records, which were the basis for the revaluation. Therefore, each ambient data record has to be identified uniquely and the new record should refer to the corresponding identifications. The ambient data structure is accordingly extended with an '*ID*' and some '*sourceID*' attributes as follows:

```
<AmbientData quality=”0.9” ID=”D56FFAA8-6D13-11d4-B675-0010F3008057”
        source=”TEMP456” time=”2003-06-24T09:30:47+01:00”>
   <weather:temperature unit=”celcius”>31</weather:temperature>
</AmbientData>

<AmbientData quality=”0.8” ID=”2C933718-CF6A-4f70-AB37-9F780B427709”
        source=”HUM987” time=”2003-06-24T09:35:32+01:00”>
   <weather:humidity>81</weather:humidity>
</AmbientData>

<AmbientData quality=”0.4” ID=”719C511C-4C40-48fe-9ACF-1224B1DF0C0E”
        source=”weatherInterpreter” time=”2003-06-24T09:36:00+01:00”>
   <sourceID>D56FFAA8-6D13-11d4-B675-0010F3008057</sourceID>
   <sourceID>2C933718-CF6A-4f70-AB37-9F780B427709</sourceID>
   <weather:weather>muggy</weather:weather>
</AmbientData>
```

In the example above, the third ambient data set was derived from the first two sets by an ambient information interpreter. The interpreter inserted the new set with the references (‘sourceID’ tags) to the base data sets, which have the ID declared as an attribute. In this way, any component, which evaluates the ambient profile, knows on which basis the information was generated. The values of the ID attribute have to be unique. They should be provided by the component, which manages the ambient information, i.e. which receives the data from the sensors and which provide the data to the interpreters and services. Whenever this component receives a new data set, it creates a unique ID and stores the data together with this ID. In the example, the UUID approach standardized by IETF [UUID] is used for the ID values. However, which approach is selected is free to the specific implementation. If the system does not assign IDs to data sets, relationships cannot be specified.

3.4.4 Ambient Awareness Functions

As depicted in Figure 3-39, ambient information is gathered from sensor networks and from the interaction with the user. The components, which gather the data, store the data into a central Ambient Information Server. This server maintains a data storage and provides the necessary interface to *store* and to *retrieve* ambient information.

To **store** data into the server, only a simple interface is needed. The source components have to create a data set according to the ambient data structure specification. This data structure contains all necessary information. The server assigns a unique identification to the data set and tracks the duration value. It should automatically delete all data sets, whose valid time is expired. This means, that the sources only have to provide ambient data sets. The ambient profile is compiled by the server based on the location data.

To **retrieve** an ambient profile, two kinds of access methods are required:

- Polling
- Subscribing

By **polling**, components asks periodically or whenever needed the server for the required information. This approach is quite insufficient; and important changes of ambient information might be missed. When the data in an ambient profile change, other components can be interested in immediate notification of this event and in receiving of the corresponding data. By supporting **subscription** mechanisms, the server can propagate ambient information by event notifications. Accordingly, the Ambient Information Server should provide an interface, which enables other component to subscribe to a specific ambient profile. Whenever, the data in the subscribed profile change, the corresponding subscribers can be notified immediately.

Furthermore, the Ambient Information Server should support the execution of interpreters. Those can also make use of the identified interfaces to store and to retrieve ambient profile data. The server should not execute the interpreter function directly, but they should be realized by external components, which then behave like ordinary clients to the server.

The Ambient Information Server has to be configurable in order to control the behavior with regard to the storage of the data. Because of the amount of data, which can be provided by complex sensor networks with a huge amount of sensor devices, the server has to be designed and to be implemented carefully. It should provide a configuration switch to activate and to deactivate the persistent storage. Usually, ambient information will always have a valid period. The server can remove all the data, whose valid period is expired. However, in order to provide some history analysis function, the access to historic, even expired, data is necessary. Such analysis function enables a better adaptation of service behavior.

As described above, the sources of ambient information have to provide their sensed data as an ambient data structure and describe them with according location area identification. As depicted in Figure 3-44, the ambient profile is created by the Ambient Information Server, which stores all ambient data sets.

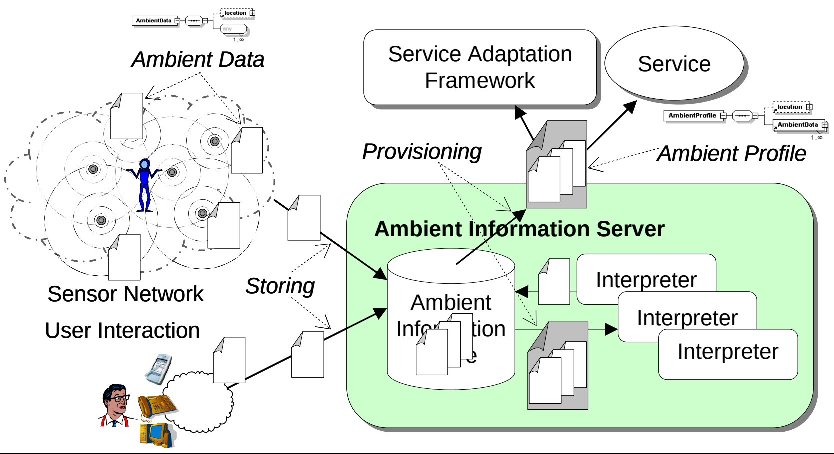


Figure 3-44: Storage and Provisioning of Ambient Data

Each ambient profile is identified and addressed by a location identifier. The location area defines for which area the data is valid. A client, which is interested in the ambient information, has to specify the target location area. The server looks for all available data, valid for the given area, and creates an according ambient profile, which is then retuned to the client. Figure 3-45 depicts how the ambient profile is compiled from the ambient data sets describing the target location area.

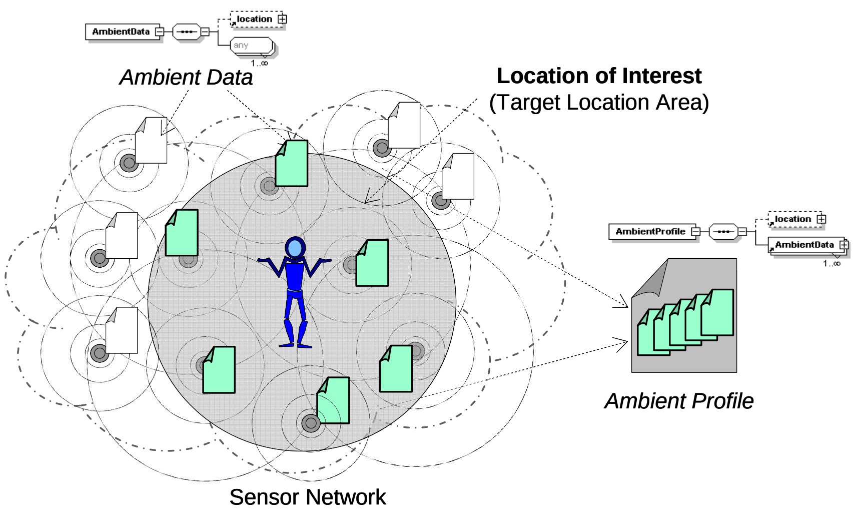


Figure 3-45: Compilation of the Ambient Profile according to Target Location Area

This means, the required functions of the Ambient Awareness model confines to the storage and retrieval. Sources of ambient information store the corresponding data sets to the Ambient Information Server. Components, which have a certain interest in available ambient information, can retrieve the available data by the retrieval functions. In this way, the interpreters can be realized by using both functions.

3.4.5 Usage of Ambient Information in I-Centric User Interaction

On the one hand, an I-centric User Interaction system should consider ambient information in the realization of the interaction. On the other hand, the system can derive and provide ambient information from the interaction itself, i.e. from the terminal used and from the reaction of the user. The general approach to derive ambient information from the user interaction is described in section 3.4.2.2. In the following, the usage of ambient information in the realization of the user interaction is described more detailed.

The usage of ambient information is relevant for the adaptation of the user interface, e.g. determination of available terminal devices and the current situation of the user. In this sense, the Delivery Context, which is introduced in section 3.5.6, has to be enriched with ambient information as depicted in Figure 3-46.

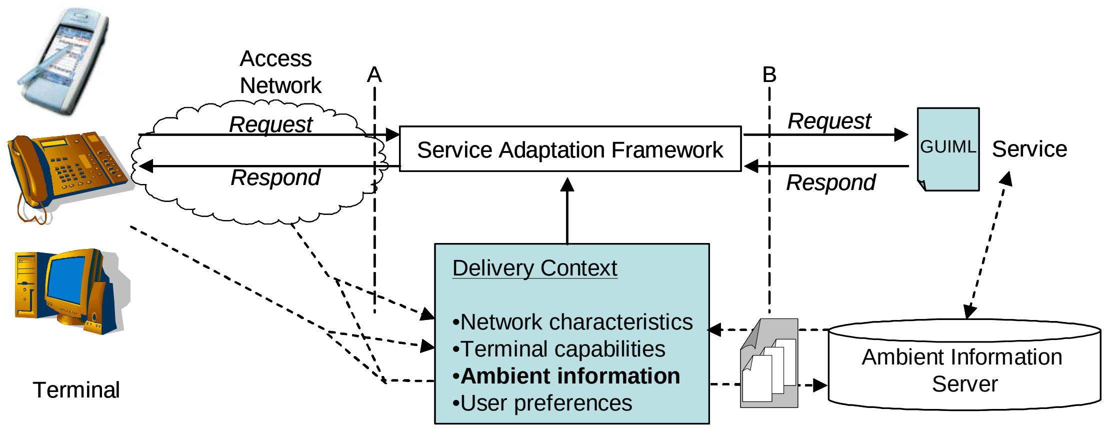


Figure 3-46: Ambient Information in the Delivery Context

Section 3.5.6 describes detailed the Delivery Context concept and introduces a language to describe the Delivery Context – the *eXtensible Delivery Context Language* (XDCL). This language is XML-based and modular structured. In order to enrich the Delivery Context with ambient information, XDCL has to be extended correspondingly. This requires the definition of suitable constructs and modules.

Basically, in XDCL a certain Delivery Context is described by a profile object. The profile can contain a number attributes, each describing a specific characteristics of the context. Modules specify valid attributes and their meaning. They are defined by using XML namespaces. In this way, XDCL can be easily extended. In order to use the data contained in an XDLC profile, all components have to know and to understand the modules used. This means, if the language was extended with a new module after a service was developed, the service cannot interpret the module attributes and cannot adapt to it appropriately. It has to ignore the corresponding attributes.

In order to include ambient information in the Delivery Context, an according XDCL module specification has to be provided. This specification is based on the developed data model (section 3.4.2). It can be applied as depicted in the following XDCL profile:

```
<profile xmlns:screen="http://www.example.org/screen-module#"
      xmlns:ambient="http://www.example.org/ambientinformation-module#">
   <screen:screenwidth>1024</screen:screenwidth>
   <screen:screenheight>768</screen:screenheight>
   <AmbientProfile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="ai ambient.xsd weather weather.xsd">
      <AmbientData time="2003-06-24T09:30:47+01:00">
         <location>
            <longitude>13.18831E</longitude>
            <latitude>52.31555N</latitude>
            <uncertaintyShape>
               <circle unit="meter">23</circle>
            </uncertaintyShape>
         </location>
      </AmbientData>
      <AmbientData quality="0.9" ID="D56FFAA8-6D13-11d4-B675-0010F3008057"
            source="TEMP456" time="2003-06-24T09:30:47+01:00">
         <weather:temperature unit="celcius">31</weather:temperature>
      </AmbientData>
      <AmbientData quality="0.8" ID="2C933718-CF6A-4f70-AB37-9F780B427709"
            source="HUM987" time="2003-06-24T09:35:32+01:00">
         <weather:humidity>81</weather:humidity>
      </AmbientData>
      <AmbientData quality="0.4" ID="719C511C-4C40-48fe-9ACF-1224B1DF0C0E"
            source="weatherInterpreter" time="2003-06-24T09:35:00+01:00">
         <sourceID>D56FFAA8-6D13-11d4-B675-0010F3008057</sourceID>
         <sourceID>2C933718-CF6A-4f70-AB37-9F780B427709</sourceID>
         <weather:weather>muggy</weather:weather>
      </AmbientData>
   </AmbientProfile>
</profile>
```

The component, which creates and maintains the Delivery Context, has to subscribe to the Ambient Information Server to retrieve the ambient information according to the current location of the user. In this way, the Delivery Context can be kept up-to-date. If the location is unknown, no ambient information can be provided. There can be systems, which apply different location schemas such as using room identifiers or zip codes. Because the ambient information data model uses the geographical data schema, such systems have to provide corresponding mapping rules or the data model has to be adapted accordingly.

As described in section 3.4.3.2, location area descriptions can exist in different formats. The geodetic representation is not user-friendly, but provides the highest accuracy. In the interaction with the user, only human-understandable presentations of location data should be used. For example, if users have to specify their location explicitly, because there is no other location information available, they should be able to do this according to their knowledge. This means, they should not be forced to input technical data, such as longitude and latitude, but should be able to use commonly applied location description schemas, such as postal addresses. Vice versa, if a service wants to present such location information, the technical presentation should be converted by the Service Adaptation Function to human-understandable presentation considering the user's preferences.

3.4.6 Conclusion

The Ambient Awareness model specifies an approach to gather, to manage, and to provide ambient information in order to enable an ambient aware behavior of services. The ambient aware behavior comprises the actual service execution, i.e. the service logic, and the realization of the user interaction, i.e. the service's user interface. The model defines mechanisms how sensed data can be provided to the system, where these are interpreted and revaluated to obtain ambient information suitable for the consideration in the service execution and the adaptation of the user interface.

The ambient information, relevant for a specific service usage and interaction, is provided through the Delivery Context to the respective components. In this way, the services but also their interaction with user can be realized in an ambient aware manner as required by I-centric User Interaction.

3.5 Generic User Interaction Model

The Generic User Interaction model, presented in the next sections, was developed based on the identified requirements in order to support I-centric User Interaction. After the general introduction, which repeats and refines the individual objectives, the basic approach is introduced. This approach is based on the idea to separate the user interface from the service logic. It defines a suitable language to describe the user interaction idependent from specific devices and user interface technologies.

The user interfaces, described in this language, are transformed to specific user interface markup languages considering the capabilities of the Access Mechanism, user preferences, and ambient information. The following sections introduce accordingly a Service Adaptation Function and a language to specify the Delivery Context, which describes the information relevant for the adaptation.

3.5.1 Introduction

The I-centric user interface, which provides the information exchange between user and objects in the individual communication space, shall be independent from specific terminal and user interface technologies. That is why this interaction model is called *generic*. However, to perform the actual interaction, the generic description has to be transformed to a concrete user interface technology and to be provided to the user's terminal.

As already stated in section 2.5.4, there is a multitude of user interface technologies and a massive number of diverse types of terminals, characterized by different capabilities. Additionally, the network connections between service and user's terminal influence the user interaction. Due to the network delays, available transfer rate, and possible congestions, the perceptible quality can be reduced. Several user interface technologies consider these characteristics in order to deliver data to the user's terminal in a best effort manner, e.g. the Wireless Application Protocol (WAP)[24] and the various video/audio streaming solutions.

To implement a service supporting even only a specific subset of available user interface technologies is a challenging and costly task. Usually, today's services and applications maintain only one user interface type, which is the most suitable according to the application domain (WWW services, telephone banking, etc.) To extend such services to support additional user

[24] [WAPARCHSP]

interface technologies is complicated and usually leads to a complete new and additional implementation with access to same underlying resources.

The Generic User Interaction model shall enable services to interact with the user through different kinds of user interface technologies without forcing the service developers to implement all the different technologies concurrently. Instead, the service developers should be supported with a Generic User Interaction technology, which can be mapped automatically to specific technologies by a corresponding adaptation function. Because services do not notice and do not have to know the specific technologies used in an interaction, this mechanism is described as *Generic User Interaction*. The according model defines necessary concepts to realize these objectives.

This fundamental approach also complies with the *Device Independence* idea of the W3C [W3CDIPRINC]. In order to realize generic and device independent user interaction, a concept to describe and define user interfaces in a generic way is needed, i.e. in a way independent from specific user interface technologies. This approach is reflected in the model by defining a *Generic User Interaction Markup Language* (GUIML).

The user interfaces and dialogs defined in the Generic User Interaction Markup Language have to be translated into technology and terminal specific representations. This translation is called *Service Adaptation*. The Generic User Interaction model has to support a *Service Adaptation Function*, which understand the Generic User Interaction Markup Language and is able to translate it to different technologies. Different kinds of user interface technologies shall be supported by the Service Adaptation Function.

According to the human senses and according to the technical possibilities, today there are mainly speech-based, tactile, and graphical user interfaces. They are fundamentally different in their nature. For example, whereas graphical user interfaces can present extensive information and many reaction (input) possibilities at once, speech-based interfaces are sequential and can therefore only present few information and few feedback possibilities. This has to be considered by the Generic User Interaction model in an appropriate way.

Generally, user interaction technologies can be classified into synchronous (e.g. speech-based) and asynchronous (e.g. message-based) communication. Accordingly, the model shall support such different natures of communication. Additionally, there has to be a support for user-initiated and service-initiated interaction, which means, that on the one hand the user can access the service actively and on the other hand that the service can contact the user, i.e. support of *pull* and *push* features.

In addition, users have specific preferences for the interaction, e.g. with regard to languages, colors, preferred communication services (to be called by phone or to be sent as a message), etc. In order to provide a most user-friendly and comfortable user interface, such preferences shall be considered in the adaptation of the generic user interface. Therefore, the Generic User Interaction model should utilize the Personalization model, which manages and provides user preferences.

Similarly, the user interaction shall be ambient aware, which means that available ambient information shall be considered in the user interaction. In some situations, certain communication mechanisms could be more suitable for the interaction, e.g. if a user is in a conversation, the contact by a message is more suitable than a phone call.

3.5.2 Overview

The user interacts with a service using a terminal device, which establishes the communication through an access network to the service. Depending on the type, the terminal supports different communication media such as speech, text, presentation of images and videos, etc. On the one hand, there are different kinds of user interaction techniques, e.g. speech versus graphical user interfaces. On the other hand, there are many technical approaches for each user interaction technique, e.g. WWW versus WAP, Public Switched Telephony Networks versus Voice over IP.

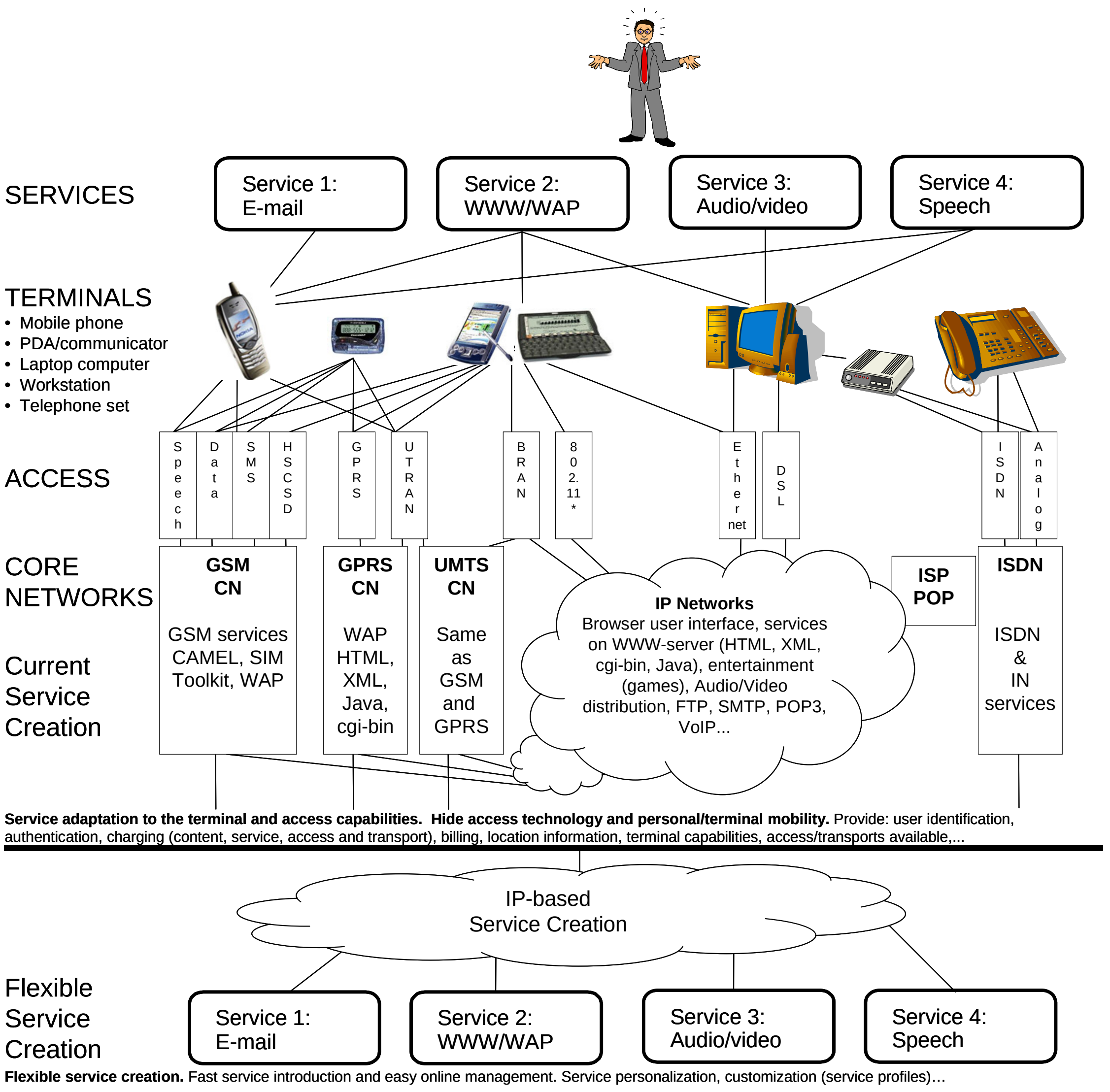

SERVICES
Service 1: E-mail
Service 2: WWW/WAP
Service 3: Audio/video
Service 4: Speech
TERMINALS
• Mobile phone
• PDA/communicator
• Laptop computer
• Workstation
• Telephone set
ACCESS
Speech
Data
SMS
HSCSD
GPRS
UTRAN
BRAN
802.11*
Ethernet
DSL
ISDN
Analog
CORE NETWORKS
Current Service Creation
GSM CN
GSM services CAMEL, SIM Toolkit, WAP
GPRS CN
WAP HTML, XML, Java, cgi-bin
UMTS CN
Same as GSM and GPRS
IP Networks
Browser user interface, services on WWW-server (HTML, XML, cgi-bin, Java), entertainment (games), Audio/Video distribution, FTP, SMTP, POP3, VoIP...
ISP POP
ISDN
ISDN & IN services
Service adaptation to the terminal and access capabilities. Hide access technology and personal/terminal mobility. Provide: user identification, authentication, charging (content, service, access and transport), billing, location information, terminal capabilities, access/transports available,...
IP-based Service Creation
Flexible Service Creation
Service 1: E-mail
Service 2: WWW/WAP
Service 3: Audio/video
Service 4: Speech
Flexible service creation. Fast service introduction and easy online management. Service personalization, customization (service profiles)…

Figure 3-47 Service Adaptation to the Terminal and Access Network Capabilities

Figure 3-47 [WWWP920] depicts an exemplary overview showing the user interacting with the services via a multitude of terminal devices, access and core network. The individual technologies shown in the figure represent only an exemplary selection of relevant technologies and do not denote the specific technologies to be supported by the model.

Generally, service developers have to choose to most suitable user interface technology depending on the type of application and usability issues. To support more than one user interface technology concurrently is a very complex and challenging task, which still cannot be solved in a satisfying way nowadays. Therefore, a large part of the service development effort is spent in the implementation of the user interface.

Existing service implementations can be extended by additional functionality to support multiple devices and user interface technologies. The advantage is that the service and its user interface are optimized for each device. However, the source code of the service implementation has to be changed or to be rewritten each time to take the individual characteristics of each device into account[25]. In terms of development and maintenance costs, this approach is very expensive and time consuming possibly leading to an inconsistency among the different user interfaces.

A more flexible approach, avoiding these disadvantages, bases on the separation of

- The user interface, and
- The internal logic of the program (service logic)

This is on the one hand a logical division, but can be also a spatial division on the other hand.

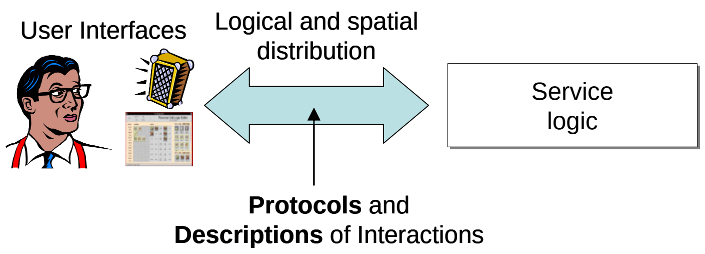

User Interfaces
Logical and spatial distribution
Service logic
Protocols and
Descriptions of Interactions

Figure 3-48: Separation of User Interface and Service Logic

Advantages of this separation are:

- An application can provide multiple user interfaces concurrently, which can be realized in different user interface technologies, e.g. speech and graphical.
- The user interfaces can be adapted to user settings and terminal capabilities homogenously. The application programmers do not have to deal with the implementation details of the user interface.
- User interface and application logic can be developed independently, i.e. user interface can be added and can be changed afterwards.
- Innovative but complex functions like multi-model user interfaces can be provided without explicit support of the service.

The Service Adaptation Function generates the technology specific representation of the user interface. For the communication between the Service Adaptation Function and the service, a suitable protocol as well as a suitable dialog description language is needed. This language describes the complete interaction, i.e. the input and output (content) parameter, independent from technical details of the terminal. The WWW concept applies this approach. The user interface description in the Hypertext Markup Language (HTML) is delivered via the Hypertext Transport Protocol (HTTP)[26] to the WWW browser application, where the graphical user interface is rendered and presented to the user.

However, this approach is limited to graphical presentation of user interfaces and is normally only support on computer workstations. Although, the WWW approach is quite flexible (especially the eXtensible HMTL (XHTML), which supports different profiles), it cannot be adopted directly for other user interface techniques, such as for speech interaction or on resource limited devices such as mobile phones.

Instead, services and applications should define their dialog with the user (input and output parameters) in a way, completely independent from device types and user interface technologies. Such generic dialog has to be translated into different specific user interface technologies

[25] Today, WAP as well as WWW services are usually optimized to the individual browser. This is necessary, because the different browsers render and present the user interfaces in different ways. This is partially due to the different capabilities of the device, but also just because of different implementations of the same standard.

[26] [RFC2616]

(WWW, speech) and to be executed on different kinds of input/output devices (terminals). In order to generate suitable user interfaces, the Service Adaptation Function has to consider the individual capabilities of the terminals, the characteristics of the network connections, the personal preferences of the user, and available ambient information.

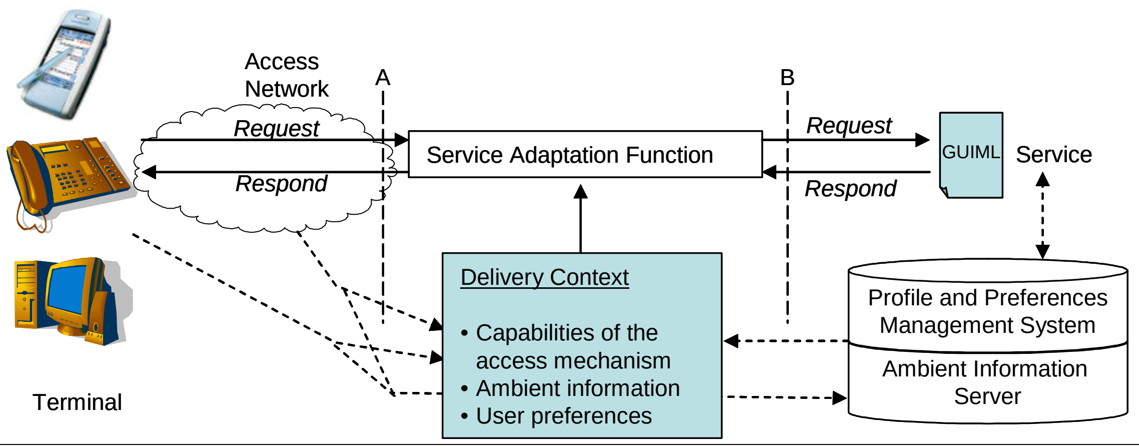

Access Network
A
B
Request
Respond
Service Adaptation Function
Request
Respond
GUIML
Service
Delivery Context
• Capabilities of the access mechanism
• Ambient information
• User preferences
Profile and Preferences Management System
Ambient Information Server
Terminal

Figure 3-49: Service Adaptation Overview

In Figure 3-49, the reference point *A* is realized by specific user interfaces technologies, such as WWW, VoiceXML, Wireless Markup Language (WML), Short Message Service (SMS), Multimedia Message Service (MMS), etc. In order to support a specific technology a mapping and corresponding transformation rules from and to the Generic User Interaction Markup Language have to be provided and to be implemented by the Service Adaptation Function. This references point also describes the capturing of the capabilities of the Access Mechanism, i.e. capabilities of terminal and network connection, as well as the capturing of ambient information, which can be derived from the user interaction.

The reference point *B* depicts the Generic User Interaction without consideration of the specific user interaction technologies. It supports synchronous, asynchronous interaction as well as user initiated and service initiated interaction. It uses a generic user interface description language, which can be transformed automatically to the supported user interface technologies (reference point A) by the Service Adaptation Function.

The Generic User Interaction model describes the Generic User Interaction (reference point B) and the mapping to specific user interface technologies (reference point A) considering the particular Delivery Context, which consists of the capabilities of the Access Mechanism, user preferences, and ambient information. The Personalization model (section 3.3) and the Ambient Awareness model (section 3.4) focus on the management and provisioning of the Delivery Context needed for the adaptation process. A service platform implementing all models jointly, provides the features, which are needed for the I-centric User Interaction. In detail, the Generic User Interaction model specifies:

- The *Generic User Interaction Markup Language* (section 3.5.4)
- The *Service Adaptation Function* (section 3.5.5)
- The *Delivery Context* describing the capabilities of the Access Mechanism (section 3.5.6)

The following sections introduce more detailed the Generic User Interaction model starting with the introduction of an approach for the generic description of user interaction.

3.5.3 Description of Generic User Interaction

A *generic* way to describe the interaction between user and service does not reflect the individual technical characteristics of user interface technologies and terminal types. This means, such a language for *Generic User Interaction* is independent from concrete technologies but can be translated into different technologies as needed. In this way, the according requirement of the idea of I-centric User Interaction can be fulfilled.

The idea of using a generic language to describe user interaction is quite related to the *Single Authoring* approach introduced in [W3CDIAC]. Although, the work of the W3C groups focuses on WWW-based services, the principles and the terms can be reused in the Generic User Interaction model for I-centric User Interaction. The most relevant principle, introduced by W3C, is the *Device Independence* (DI). According to [W3CDIPRINC], the goal of this principle is that WWW content and applications can be authored, generated, or adapted for a better user experience when interacting with presentations via many different WWW-connectable Access Mechanisms. The general phrase ‘Device Independence’ is used for this, although the Access Mechanisms can include a diversity of devices, user agents, channels, modalities, formats, etc.

Analyzing the existing technical approaches and the human-machine interaction in general, two different aspects of a user interface description can be identified, concerning:

- The dialog
- The content

The dialog of an interaction describes the logic and the structure of information to be exchanged, as well as the possible reactions (inputs) of the user. Whereas in human-human interaction the dialog is dynamic, in human machine interaction this dialog is usually static, i.e. restricted to optional choices provided by the service. All possibilities have to be covered by the dialog designer. Basically, all available options of reaction are presented to the user, e.g. menus, selection lists, links, etc.

Today, first applications like advanced telephone-banking systems also accept natural input without giving fixed input alternatives. The input is analyzed to extract reasonable keywords. Based on the recognized keywords and the current context (e.g. preceding inputs) the most suitable selection is done. However, this approach relies on intelligent and complex algorithms, which can lead to misunderstandings and wrong behavior. Because of this, existing applications are still too error-prone and therefore only used in highly specialized applications today.

Beside the dialog, also content has to be exchanged in an interaction. Services provide content in various media types (text, images, audio, etc.) Depending on the application scenario, users may have to provide content, e.g. voice input, which is to be processed by the service. In a device-independent interaction, the multimedia content must be adapted to capabilities of the terminal used. This includes media type and media format conversion. According concepts are discussed in [PFEIFER1].

This means, the Generic User Interaction description has to provide suitable mechanisms to describe the dialog on the on hand and to describe multimedia content, which is exchanged in the interaction, on the other hand. Both together, i.e. the dialog description language and multimedia content description language, define the Generic User Interaction Markup Language. The following two sections introduce suitable concepts for each in detail.

3.5.3.1 Dialog Description

For the generic description of dialogs, the XForms standard [W3CXFORMS] can be applied. This standard provides a platform-independent markup language to describe dialogs and is under development by the W3C. Content, presentation, and logic are strictly separated through the concept of the *XForms Model* and the *XForms User Interface.* Flexible presentation options can be attached to the form definition as shown in Figure 3-50.

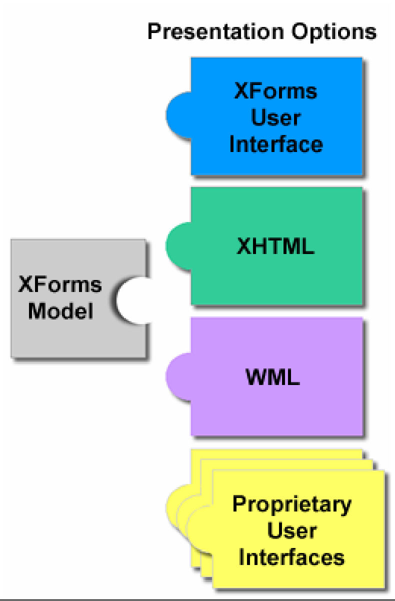


Figure 3-50: XForms Model with a Variety of User Interface Technologies (Presentation Options)

This means, the generic dialog can be described in an XForms Model. The actual representation in a specific user interface technology, such as HTML, WML, or speech-based user interface, is defined the by XForms User Interface. In this way, the requirement of the Generic User Interaction model to describe dialogs in a platform-independent manner can be fulfilled by XForms.

The XForms user interface provides a standard set of visual controls for designing dialogs. The controls can be used inside any other XML[27]-based languages. The collection of form data is defined in the XForms model and expressed as XML instance data. Thus, complete XML structures are submitted in XForms instead of flat name-value-pairs known from XHTML submissions. The *XForms Submit Protocol* [W3CXFORMS] defines how XForms send and receive data.

The full XForms concept offers advanced functionality that small mobile devices are not capable of, like validation of user input on the client side, i.e. within the browser. There is also a conformance level, which is suitable for limited devices. It is called *XForms Basic* [W3CXFORMS]. It uses only a subset of XML Schema[28] and does not include any resource-intensive features. It is possible to define a hierarchy of form controls. It can be used in the dialog adaptation process to define parts of the dialog, which should not be separated into multiple pages.

XForms themselves do not describe the content of a dialog. Instead, they are to be integrated in other XML-based languages, like XHTML or SMIL. XForms Basic meets the requirements of the dialog description for the Single Authoring language, required to realize I-centric User Interaction. The definition of this subset of XForms enables the application also on limited devices, such as PDAs or mobile phones.

3.5.3.2 Multimedia Content Description

To enable the simple authoring of interactive audiovisual presentations, the W3C has developed the *Synchronized Multimedia Integration Language* (SMIL) [W3CSMIL]. As described in [W3CSMIL], SMIL is an XML-based language and typically used for authoring “multimedia presentations, which integrate audio and video with images, text, or any other media type”.

[27] XML: Extensible Markup Language [W3CXML]

[28] [W3CXMLSCH0, W3CXMLSCH1, W3CXMLSCH2]

SMIL can describe the temporal behavior of multimedia presentations, references (hyperlinks) to other media elements, as well as the layout of individual media components.

SMIL 2.0 [W3CSMIL] is an evolution of the SMIL 1.0 [W3CSMIL1] standard defining a set of markup modules, which specifies the semantics and according XML syntax for additional functionality. It was an important design goal of SMIL 2.0 to allow reusing of SMIL syntax and semantics in other XML-based languages, e.g. to integrate timing into XHTML documents [W3CXHTML]. This idea is based on the concepts of *modularization* and *profiling* [W3CXHTMMO]. To understand the concepts, a very short introduction is given in the following.

A '*module*' is a collection of semantically related XML elements, attributes, and attribute values, which form a unit. The elements of a module are coherent through their common namespace. A '*language profile*' is a combination of atomic modules. Commonly, a main language profile incorporates nearly all the modules associated with a single namespace. The SMIL 2.0 language profile uses most of the SMIL 2.0 modules. Language profiles can also be included in other language profiles and can describe combinations of modules with different namespaces. XHTML+SMIL [W3CXHTMLS] is an example of a combined language profile. A '*structure module'* contains the root element of the language profile, e.g. '<html>'. The '*host language*' characterizes the core of the functionality provided by the namespace. It must incorporate the 'structure module'.

SMIL 2.0 Basic [W3CSMIL] is a language profile, which meets the needs of resource-constraint devices such as mobile phones. Each platform has its own capabilities. Thus, not all features of SMIL 2.0 will be required on all platforms. SMIL Basic only includes a subset of the SMIL 2.0 modules and is host language conformant.

SMIL functionality is fragmented into ten functional areas while each functional area is further partitioned into modules. These modules are associated with the SMIL namespace. The most relevant functional areas for the description of media elements and content control are:

- **Media Objects functions** allow the inclusion of media elements into XML-based documents and to provide a further description of these elements. SMIL classifies media elements into animations, audio, image, text, text-stream, and video.
- **Content Control functions** provide runtime content choices and optimized content delivery. Thus, SMIL allows specifying how the presentation is adapted according to different contexts.

SMIL 2.0 is already an accepted industry standard. The RealOne Player [WWWREALPL], the Apple Quicktime Player [WWWQUICKT], and the Internet Explorer 6.0 [WWWMSIE] support it. In addition, the use of SMIL modules has already gained acceptance in the Third Generation Partnership Project [3GPP-TS26.234]. SMIL is used for the presentation of the Multimedia Messaging Service (MMS) [WAPMMSARC], too.

3.5.4 Generic User Interaction Markup Language

This section specifies the *Generic User Interaction Markup Language* (GUIML)[29]. This language supports the Single Authoring concept and bases on W3C standard, namely XHTML, XForms, SMIL, and CCS. The Document Type Definition (DTD) for GUIML can be found in the appendix (section 7.1). In this sense, the requirement to reuse existing standards instead to develop a new user interaction technology as described in section 2.1 is fulfilled. GUIML applies a selection of public standards resulting in a powerful technology to describe user interaction in a device independent, i.e. generic, way. Each selected standard covers a specific area:

[29] An initial approach of GUIML was introduced in the diploma thesis [MROHRS].

- XHTML to describe the general structure
- XForms to describe dialogs
- SMIL to describe media elements
- CSS to describe layout

Accordingly, a GUIML document can be specified by using the syntax of the individual language, defined above. This means, each GUIML document consists of XHTML, XForms, SMIL, and CSS tags. There are no further extensions defined by GUIML. In this way, GUIML documents can be processed and created by all component, which understand XML and are able to validate an XML document against a DTD or XML Schema specification.

The selection of these standards was based on the results of evaluations, which were done in several research projects. Of course, there are also further alternatives for each. For example, to describe media elements, the MPEG7 [ISOMPEG7, HUNTER] and MPEG21 [ISOMPEG21] standards are very competitive alternatives. SMIL was chosen because it comes from the W3C like the other selected standards and because it suits very well in the W3C's Device Independence approach [W3CDIPRINC].

Only some XHTML features can be used easily for all different kinds of terminal devices without causing rendering problems. These include the use of text and basic text formatting, such as standard headings, paragraphs, and lists. Basic tables cannot be used for design or layout, but for their original intent: tabular data formatting. Standard hypertext links are also offered because this is needed for XHTML to be a valid host language [W3CXHTMMO], but they should not be used. Instead, the XForms controls, described below, should be used because they allow a better dialog adaptation. The XHTML Basic forms and images are not included in this specification because XForms and SMIL provide a better support and hence are more suitable to fulfill the requirements.

3.5.4.1 Describing User Interactions

As described in section 3.5.3.1, GUIML uses the XForms Basic module to describe the dialogs. Thus, a subset of the complex XForms standard appropriate for the adaptation to small mobile devices is provided in GUIML. The XForms contain a section, which describes what the form does, called the XForms model, and another section, which describes how the form is to be presented, the XForms user interface or presentation. The XForms user interface defines a device-independent, platform-neutral set of form controls suitable for general-purpose use. The user interface is bound to the XForms model via the XForms binding mechanism. The following pattern shows an exemplary XForms model:

```
<xfm:model id=”menu”>
   <xfm:submission action=”loginAction.jsp” id=”submit” method=”get”/>
   <xfm:instance>
      <selection/>
      <somedigits>123</somedigits>
   </xfm:instance>
</xfm:model>
```

By referring to the respective XForms model, the presentation is specified by the XForms user interface, as depicted in the following example:

```
<xfm:input model=”menu” ref=”somedigits”>
   <xfm:label>Please enter some digits</xfm:label>
   <xfm:help>Just enter some digits</xfm:help>
</xfm:input>
<xfm:submit submission=”submit” model=”menu”>
   <xfm:label>Next Page</xfm:label>
   <xfm:help>Next Page</xfm:help>
</xfm:submit>
```

The advantage of this design is that the user interface is not hard-coded, which means that such user interface can be transformed to the specific user interface technologies and adapted to device capabilities accordingly. The form controls can be rendered with associated captions appropriately. This simplifies the markup description of form controls.

The XForms model contains submit information, i.e. the action attribute hold the address to call when the form is submitted. This attribute can be set to 'GET' or 'POST'. The form can specify an identifier to be included the submit information, which can be used in the service logic to identify the submitted request.

Furthermore, the XForms model defines instance variables. All variables that are used in the form have to be specified in the form declaration. The types of the variables can also be specified. The value of instance variables can be preset to a default value. The example above uses a predefined variable 'somedigits' with its value set to '123'. The possible values of a variable, which is used in a 'selectMany' control, must be separated by a space.

The powerful concept of XForms to submit data as complete XML structure is not supported by today's WWW browsers, which submit data as simple name value pairs only. Thus, the flexible XForms instance model cannot be used for these devices as it is. In GUIML, the usage of form instances is supported, but it only allows the definition of direct child elements, i.e. only one nesting step. The instance variables itself cannot have further child elements. Although this is a compromise, it allows the service author to separate the data description from presentation and helps the adaptation function to render the form properly, i.e. to create valid HTML, VoiceXML, or WML forms based on the XForms declarations. Table 3-5 lists the XForms form controls supported by GUIML and gives a short description. The more detailed description of each form control can be found in the appendix in section 7.6.1.

Element	***Description***
input	Enables free-form data entry
secret	Used for entering information that is considered sensitive, and thus not echoed to a visual or aural display as it is being entered, e.g. password entry
text area	Enables free-form data entry and is intended for use in entering multiline content, e.g. the body of an e-mail message
trigger	This form control may be used to realize submits or links by using XForms actions
submit	Initiates submission of all or part of the instance data to which it is bound
select1	Allows the user to make a single selection from multiple choices – the choices can contain actions, thus this control may be used to realize menus with multiple links
select	Allows the user to make multiple selections from a set of choices

Table 3-5: XForms Form Controls supported in GUIML

The XForms 'group' instruction allows the grouping of dialog elements that belong together. It is used to specify that the form should not be separated into multiple segments by the adaptation process. This is necessary for forms, which should be filled out coherently (e.g. an e-mail message with the various fields for address, subject, and body).

It might be possible that the XForms features supported by GUIML are not sufficient for certain needs to realize specific user interfaces. The advantage of the supported subset is that it does not impose complex browser capabilities on the target devices. If specific features are additionally needed, they can easily be integrated to the GUIML specification on demand. The features, selected for the presented specification, base on a careful evaluation of the capabilities of the targeted terminals and user interface technologies (WWW, WAP, Voice Interaction).

3.5.4.2 Describing Media Elements

The XHTML+SMIL profile defines a set of XHTML abstract modules that support a subset of the SMIL 2.0 specification. It includes the functionality of SMIL 2.0 modules providing support for animation, content control, media objects, timing and synchronization, and transition effects. The profile also integrates SMIL 2.0 features directly with XHTML and CSS, describing how SMIL can be used to manipulate XHTML and CSS features. Additional semantics are defined for some XHTML elements and CSS properties.

GUIML adopts the *MediaAccessibility* and *MediaDescription* modules of the *SMIL Basic Media Objects* functionality [W3CSMIL] to describe media elements. To select the most appropriate media element at runtime, the *CustomTestAttributes* module of the *SMIL Basic Content Control* functionality is supported into GUIML, too. Using this module, the generic user interface can specify different alternative media elements, each suitable for different device classes, to avoid an automatic conversion. In this way, the user interface can provide the content in different media representations, e.g. as text, image, or audio. According to the evaluation of the test attribute, the most suitable alternative is selected for presentation. Table 3-6 presents all SMIL media elements supported in GUIML.

Elements	***Description***
ref	Generic media reference
audio	Audio clip
img	Image, such as GIF or JPEG
text	Text reference
video	Video clip
animation	Animation, e.g. Shockwave technology

Table 3-6: SMIL Media Object Elements supported by GUIML

In principle, all of media elements as shown in Table 3-6 are semantically identical. It is actually not necessary to use the particular media object element for inserting media elements, e.g. the generic reference can be used always. This is possible, because the exact type of the media object can be derived from the type of the media object element itself, such as the type information contained in the type attribute or the suffix of the filename. However, the suitable media object element name should be used, if the type of the object is known. This increases the readability of the document and ensures the compatibility to future implementations and changes, likewise. The media object elements can have the several attributes as depicted in Table 3-7.

Attribute	***Description***	
src	The URI of the media element, used for locating and fetching the associated media	
type	The content type of the media objects, referenced by the ‘src’ attribute. This attribute should be the first choice for the Service Adaptation Function to determine the type of the media object. The following content types are recognized (MIME-type and corresponding file extension):	
	image/gif	.gif
	image/jpeg	.jpg
	image/vnd.wap.wbmp	.wbmp
	video/avi	.avi
	video/mpeg	.mpg
	video/x-ms-wmv	.wmv
	audio/wav	.wav
	audio/x-ms-wma	.wma
	audio/x-au	.au
	application/x-shockwave-flash	.swf
	text/plain	.txt, or data:…

Attribute	***Description***
alt	A short textual alternative, which can be presented to the user instead of the media object, if the user agent cannot present it, e.g. if the media object is an image, a voice-based user interface can use this text instead. It may also be displayed in addition to the media. The value of this attribute is a text string.
longdesc	A link (URI) to a long (textual) description of the media object
abstract	A brief description of the content contained in the element, i.e. a summary. It is typically to be used in an overview, e.g. table of contents.
author	The name of the author of the content contained in the element
copyright	The copyright notice of the content element
title	The title of the media object

Table 3-7: Attributes of the SMIL Media Object Elements used by GUIML

Media objects are usually inserted into the document by providing a reference, described by a Universal Resource Identifier (URI) [RFC2396], in the 'src' attribute. However, the media objects data can be provided alternatively inline by using the 'src' attribute value starting with 'data:' followed by text data or base64 encoded binary data. This is useful for small media objects primarily.

Providing Alternatives of Media Elements

By using the SMIL switch statement, GUIML supports the specification of a set of alternative media elements from which only the most suitable element is chosen. This is reasonable, if an automatic adaptation of the content is not possible or does not produce sufficient quality. Using alternatives, the services themselves can provide suitable alternatives. For example, if the service provides a handsome animation using shockwave technology, it can provide also a non-animated (still) image, because not all clients are able to process shockwave media. In the following example, a weather service provides a weather report using different media alternatives: handsome shockwave weather map, a rendered picture, and pure text:

```
<smil:switch>
   <smil:ref src=”weather_report.swf” />
   <smil:ref src=” weather_report.jpg” />
   <smil:ref src=”data:18°C, bewölkt, Niederschlag”
             systemLanguage=”de” />
   <smil:ref src=”data:64°F, claudy, rainfall” systemLanguage=”us”/>
</smil:switch>
```

In this example, one of the provided media element is to be selected from the list of four alternatives, each described by a '<smil:ref>' tag. The Service Adaptation Function evaluates the types of the different media elements to decide which media object to choose. The example above does not use any 'type' attributes. Therefore, the actual type of each alternative is derived from the file extension. For the last two alternatives, which use inline data, the type 'text' is assumed. The first media element, the actual target system (terminal) is capable to present, is chosen, i.e. for WWW browsers the shockwave animation or the JPEG image, for voice interaction or simple text messaging the text data.

In addition to the type of media objects, it is also necessary to evaluate if the terminal is capable of presenting the media elements with respect to their properties. For each alternative, the respective properties have to be analyzed and compared to the capabilities of the Access Mechanism, likewise. For example, the resolution of an image is 400x300 pixels, but the screen size of the actual terminal supports only 80x40 pixels (typical for a mobile device). If no suitable image is provided, the Service Adaptation Function has either to resize the image or to choose a different alternative, possibly of different type. It depends on the implementation and the supported media conversion functions, which alternative should be used. If an automatic conversion of the image is not possible, a different perhaps textual alternative should be preferred. The applied algorithms have to define their own thresholds in order to provide the best look and feel to the user.

Beside the terminal capabilities, there are further important factors for the selection of suitable media elements, such as the user's preferences. The user might not want to see images at all or might prefer certain languages. The example above provides two language alternatives for the text representation. The selection algorithm uses the 'systemLanguage' attribute to determine the suitable alternative to be displayed to the user.

All the information, to be considered in the selection process, is summarized in the Delivery Context (see section 3.5.6). Beside, GUIML specifies some general attributes to be used by the switch statements, denoted as *Adaptation Switches*.

3.5.4.3 Adaptation Switches

As explained in previous section, certain attributes can be added to media object elements to control the decision of selection process. These attributes are denoted as *Adaptation Switches*. They may be added to all elements of a GUIML documents in order to include or to exclude specific parts of the content depending on the specific conditions of the individual switches. Table 3-8 presents the individual Adaptation Switches supported in the current specification of GUIML.

Attribute	***Description***
bitrate	The approximate bandwidth, in bits-per-second, available to the system
language	A comma-separated list of language names as defined in [RFC1766]
screenDepth	The color depth of the screen palette in bits
screenSize	Screen size in pixels, 'screen-height'X'screen-width'
deviceClass	One of HTML, WML, or VXML– checks the target user interface technology to enable specific content selection
systemCapable, systemNotCapable	One of: color, image, video, soundOutput, graphics These switches enable a more user agent specific content decisions. For example, if the user agent is capable of playing audio, e.g. a VoiceXML user agent, but no video data, the video has to be converted to audio (by extracting the audio data). In order to avoid the automatic extraction process (because quality is not good enough), the service can specify alternatives manually with the switch statement: systemNotCapable = 'video'.

Table 3-8: The Adaptation Switches supported by GUIML

GUIML can be extended easily to support additional attributes. The Adaptation Switches are dependent on the availability of data as described by the Delivery Context (section 3.5.6). A media object is selected when the given test attributes (conditions) are successfully evaluated as described by following example:

```
<smil:switch>
   <smil:ref src=”data:deutsch” systemLanguage=”de” />
   <smil:ref src=”data:english” systemLanguage=”en” />
</smil:switch>
```

In the example above, the first alternative is chosen, if the required language is German, possibly specified by user preferences or by terminal settings. The second alternative is a presentation in English. If the Adaptation Switches do not fit to the request, e.g. French language would be requested, the Service Adaptation Function has to decide whether not to present the content at all or to select a default element, which could be the first element of the switch statement. This decision can be controlled by the user settings, e.g. ‘Languages=de,en,us’.

3.5.4.4 Support of Frames for Complex Sites

Frames are a commonly used technology in WWW in order to present web pages in a more user-friendly way. Usually, frames are used for headers and footers (top-level menus), for detailed menus (displaying the navigation structure), and for the actual content to be presented. Frames simplify the navigation even through complex structured documents. GUIML provides a similar support for frames. However, because of the different presentation capabilities of the user interface technologies, special attention must be paid to the support of frames in order to ensure the Device Independence.

The site layout, i.e. the definition of special frames and their purpose, has to be configured for each device class, the Service Adaptation Framework supports. This is necessary, because menus have to be realized in speech-driven user interfaces in a different way than in WWW-based user interfaces. For instance, WAP has a special support for menus, e.g. the menu button, which opens the menu on the WAP terminal. This means, menus have to be separated reasonably from the actual content to be displayed. GUIML defines some common frames (‘main’ and ‘menu’); and the site layout configuration specify how these special frames should be realized in the particular device classes, i.e. for HTML, WML, or VoiceXML respectively. The GUIML documents should refer to these special frames in order to present menus and content. They can use additional frames, which are not declared in the site layout configuration. Such frames will be translated into the specific user interface as good as possible, for instance by eliminating the frame structure, if frames are not supported like in VoiceXML. Section 7.6.2 describes an exemplary layout configuration set, which transforms frames into a table-based representation.

Two identifiers have a special meaning. For devices, which can only present one frame at a time, e.g. WAP and VoiceXML browsers, only the frame with the identifier ‘main’ is presented. The frames, which contain a menu indicate by the ‘menu’, cannot be accessed directly from VoiceXML browsers. Instead, all links in ‘menu’ frames have to be presented to the user indirectly, i.e. on a separated page. In this way, the user can navigate through the different menus and content.

In the GUIML document, frames are referenced as follows:

```
<document>
   <head>
      <title>…</title>
   </head>
   <body>
      <frame id=”menu” href=”menu” title=”Menu” style=”…”/>
      <frame id=”submenu” href=”submenu” title=”submenu” style=”…”/>
      <frame href=”main” id=”main”/>
      <frame href=”main2” id=”main2”xml:base=”http://www.server.com”/>
   </body>
</document>
```

The Service Adaptation Function copies the contents of the frame elements into the table structure according to the specified identifier. The title attribute assigns the frame a name that is presented to the user to select a new frame. The style attribute can be used to assign cascading style sheets to the individual frames. If a frame should be loaded from another URL, the ‘href’ attribute can be used. The referenced document has to define a complete frame including the frame element with the corresponding identifier.

3.5.4.5 CSS Page Layout

The GUIML language itself does not provide elements to layout pages. GUIML is based on XHTML Basic, which aims at supporting the service provider to design the flow of information in the page and not to layout the page. Tables in XHTML Basic can only show tabular data. They cannot be used to arrange individual elements. Thus, GUIML documents can be rendered on devices that require an information flow, like WAP terminals.

To be able to design handsome HTML pages, CSS can be used. A reference to a CSS file in the configuration file, described above, has to be specified. Each GUIML element accepts 'style' and 'class' attributes, too. It is reasonable to use a separated CSS configuration file to support future devices, which eventually will support other CSS attributes. Further documentation can be found at [W3CCSS2SPEC, WAPCSSSP].

The flow of information has to be kept consistent also when using CSS for the page layout. The terminal may not support the positioning of elements and present them in the order as they appear in the GUIML document. Then, only the reasonable structure of the information flow enables assures that the user can understand the content.

3.5.4.6 General Remarks

To create GUIML documents that can make optimal use of the features of the target terminal device, the author has to follow some rules. Although GUIML offers the possibility to build HTML like user forms, user interactions should not be designed that way. In contrast to the most WWW-pages today, which for example use tables for layout purposes, the individual GUIML elements shall be used, according to what they are designed for, in order to enable the reasonable automatic adaptation.

For example, if a list of names, which can be edited or deleted, should be presented to the user. In the usual (WWW-based) way, the author would use a table, whose first column holds the name. The second column has an edit link and the third column contains a delete link. That is a reasonable solution for a HTML page. However, how could such a page be transformed automatically to the format of other terminal devices, i.e. to WAP or to VoiceXML? There is no possibility for the transformation engine to make the best use of the terminal device features, because the tables' cells have no association to each other. It could only transform the table to a mobile device using a WML table, which would be too wide and too long for the small display.

A better solution is to use the XForms control *select1* and two XForms buttons as provided by GUIML. All the names are items of the *select1* control, and the two buttons submit the form. That way, the transformation engine can render the list of names as radio buttons for the WWW browser and as drop down list for a WAP browser.

When GUIML documents are designed, optimal use of the elements, the markup language offers, should be made. The documents should contain neither long texts and lists, nor many links. The presentation of a page should not be designed by the author directly. It is the task of the transformation engine, i.e. the Service Adaptation Function, to create the presentation of the page. Tables, as stated above, should be used only for tabular data presentation and not for the positioning of media elements. The problem of rendering the page for small devices must always be in ones mind. Generally, when designing GUIML documents, the different results of the transformation process have to be evaluated always, whether they correspond to the requirements and whether they provide a suitable user interaction. According developments tools should support the service developer in doing this.

3.5.5 Service Adaptation Function

The process of adapting the single authored service to terminal device specific formats considering the Delivery Context is called Service Adaptation. This approach is mentioned in [W3CDIPRINC]. To support Service Adaptation, a framework that performs the adaptation process is necessary, which is denoted as the *Service Adaptation Framework.*

“The authoring burden could be minimized through clear separation of business logic (application-oriented functionality), interaction logic, adaptation rules, and customization metadata. These would form the input (expressed in some intermediate representation) to an automatic adaptation process.” [W3CDIPRINC]

The adaptation process has to adapt the user interface of services to the capabilities of the different access by producing multiple user experiences. The automatic adaptation process consists of different types of adaptation, which are briefly described in the following.

[BEKKUM] distinguished two important types of adaptation based on *what* is adapted. This is called ‘*adaptation aspect*’:

- **Content adaptation:** The change of the content based on the Delivery Context belongs to this category, i.e. selecting and converting media elements.
- **Logic adaptation:** Altering the logic of the service interaction is much more complex, but absolutely required due to the different characteristics. The dialogs eventually have to be executed in multiple, separated steps, e.g. on small-screen devices dialogs are separated to span multiple pages. Complex pages with multiple frames might have other navigation logic when accessed with different devices.

Furthermore, [BEKKUM] identifies a distinction in the factor *when* the adaptation takes place. This is called ‘*adaptation time*’:

- **Static adaptation:** The adaptation is done at the start of the service. This is the more basic and common form of adaptation, but only suitable if the circumstances are relatively stable. If just a complete WWW page should be presented to the user where the content does not change when the Delivery Context changes, this adaptation is applicable. In the more dynamic context of mobile users, less static type pf adaptation is required, an addition.
- **Dynamic adaptation:** The service is adapted when it is executed. An example is the change of the rate of a video stream if a mobile user roams from a high-bandwidth to a low-bandwidth network. In this case, the adaptation process does not end when the playback of the video starts. The dynamic adaptation of the video stream is necessary as long as the video is running.

Current methods of WWW content adaptation use style sheets to convert an XML-based intermediate representation to device specific XML formats. Devices, which do not use XML-based languages to describe their content, need a further conversion or gateways that support this conversion, such as a VoiceXML gateway. Because the capabilities of the terminal devices differ, the adaptation of content to different styles is not enough. Additionally, a conversion to different modes is necessary [WU]. A media element can have more than one mode of representation, e.g. a text and an audio representation.

Basically, the Service Adaptation Function translates the content described in GUIML into the specific user interface markup language required to present the content on the target terminal. For example, this means to convert the GUIML document into an HTML, WML, or VoiceXML document. These three different markup languages were chosen for the prototype implementation, because they all together depict all relevant differences and peculiarities of today available and relevant terminal technologies (i.e. speech interaction, graphic interaction, small limited displays, and big displays). Because of the modular design, the Service Adaptation Function can be easily extended to support also further user interface technologies, e.g. tactile user interfaces, by providing translation mechanisms to the according markup languages.

First of all, the Service Adaptation Function processes the Adaptation Switches, which are specified in the GUIML document. Accordingly, the resulting document depicts the dialog and the content to be presented on the user’s terminal. The remaining translation of the GUIML document into the target markup languages consists of the adaptation of:

- The dialog (navigation)
- The content including media adaptation and conversion

The **dialog** has to be translated to the navigation features supported by the target user interfaces. The available navigation features differ in the particular user interface technologies because of the fundamentally different nature. For example, in a graphical user interface, the user can easily handle a multitude of feedback possibilities, e.g. forms and links. This is commonly used in today’s web portals, where the user has plenty input possibilities. In speech-based interfaces as well as in user interfaces on mobile terminals, e.g. WAP, the navigation has to be simple, because of the sequential interaction.

This means, the speech interaction has to follow a simple structure. The dialog should consist of a small number of answer possibilities easy to grasp by the user. All answer options have to be provided to the user — one after the other, i.e. “*select 1 for service1, select 2 for service2, ..., select 0 for help*”. Whereas users can easily deal for example with three different options, more than 10 options are already to complex. The dialog adaptation maps the navigation described in the GUIML document into the targeted user interface technology. This can imply that the navigation is restructured according to the capabilities of the user interface technology. The restructuring can split a single GUIML document into multiple target documents, i.e. several WAP or VoiceXML pages, while keeping the general navigation structure.

The **content** adaptation adjusts the content to be presented to the format as required by the targeted user interface. This includes also the adaptation of media elements using *media format* and *media type* conversion, e.g. GIF to JPEG conversion as well as text to speech translation. The Service Adaptation Function has to determine the type of the media object to decide if the actual Access Mechanism is capable to present it. Therefore, it analyses the properties of the media elements as depicted in Figure 3-51 and checks if the type attribute containing the content type is available. If not, the file name extension of the src attribute is has to be evaluated. If the extension is unknown to the system, perhaps content negotiation helps to determine the media object type [LEMLOUMA]. The service provider has to make sure that the Service Adaptation Function is able to determine the type of the media object, preferably by providing the type attribute.

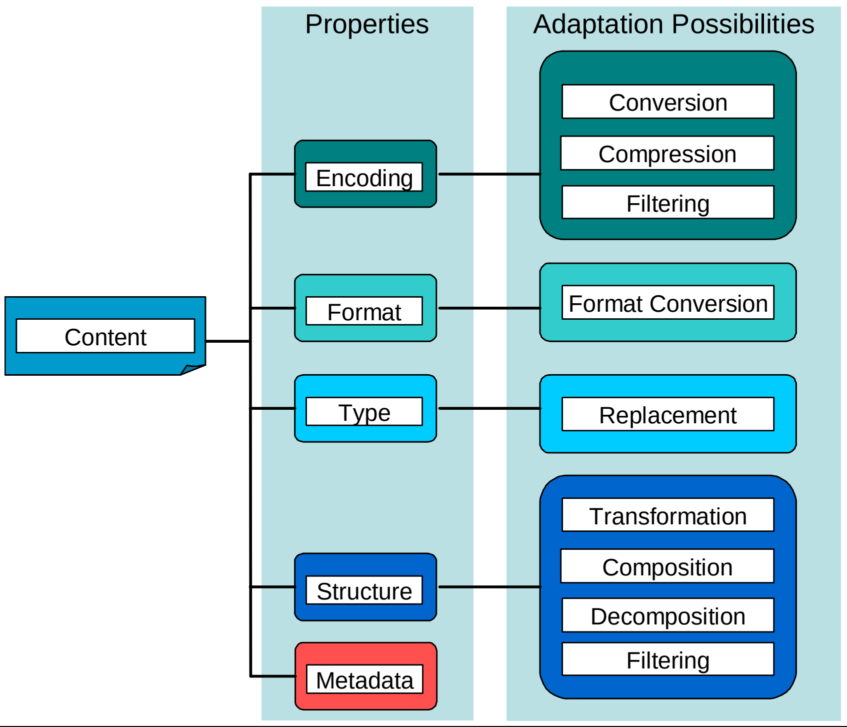


Figure 3-51: Content Adaptation Overview

The media element selection is done in two steps. In the first step, the Service Adaptation Function tries to find a media element that the user agent is able to present. When a list of alternative media elements exists, it is sequentially traversed. If no suitable media element can be found in the list of alternatives, step two follows.

The Service Adaptation Function determines the conversion possibilities of all media elements, i.e. whose presentation can be achieved through conversion. Then, it checks again if any of the converted presentations would fit to the requirements of the targeted user interface. If a conversion strategy is found, the Service Adaptation Function inserts a reference to a *media converter* with the source address and necessary conversion parameters into the document. That way, the media converter is requested when the document is loaded and returns the media element adapted to the device specific capabilities. If no suitable media element is provided directly and cannot be provided by conversion, the Service Adaptation Function has to use alternative descriptions, which should be specified by the 'alt' attribute, 'longdesc' attribute, or 'abstract' and 'title' attributes. The service author should ensure that sufficient information is provided to the Service Adaptation Function in order to enable a successful adaptation.

The Service Adaptation Function has to categorize all media alternatives with regard to the semantic significance. Because conversion means usually a loss of information, e.g. if a colorful image is converted to a small black-white image, the Service Adaptation Function should first try to use the media alternatives as provided by the GUIML document. Only, if no suitable media alternatives are found, the conversion should be applied instead. The media type conversion means the biggest loss of semantic information. Fundamentally, the degree of semantic information loss depends on the quality of the individual converter implementation and algorithms used.

The following examples illustrate the media element selection and media object conversion process. Figure 3-52 shows three alternative media elements – a big image, a small image, and a textual representation – described in according SMIL notations:

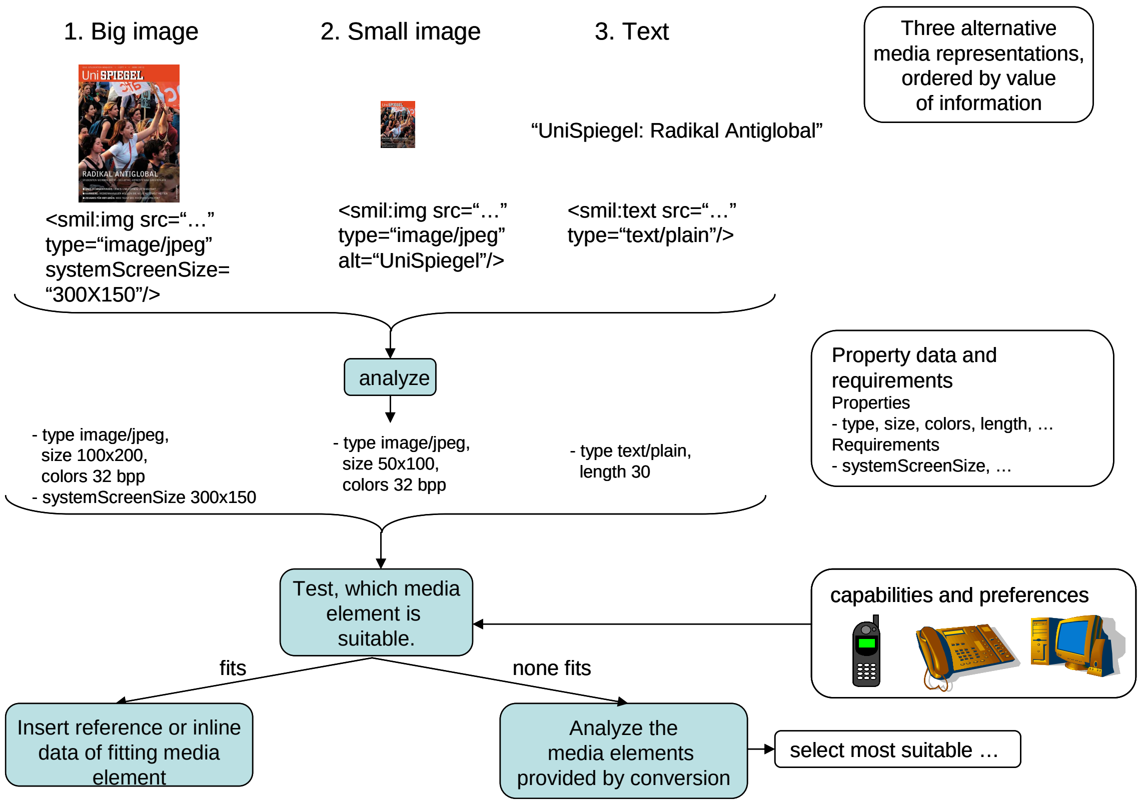

1. Big image
2. Small image
3. Text
Three alternative media representations, ordered by value of information
“UniSpiegel: Radikal Antiglobal”
<smil:img src="…" type="image/jpeg" systemScreenSize="300X150"/>
<smil:img src="…" type="image/jpeg" alt="UniSpiegel"/>
<smil:text src="…" type="text/plain"/>
analyze
Property data and requirements
Properties
- type, size, colors, length, …
Requirements
- systemScreenSize, …
- type image/jpeg, size 100x200, colors 32 bpp
- systemScreenSize 300x150
- type image/jpeg, size 50x100, colors 32 bpp
- type text/plain, length 30
Test, which media element is suitable.
capabilities and preferences
fits
none fits
Insert reference or inline data of fitting media element
Analyze the media elements provided by conversion
select most suitable …

Figure 3-52: Alternative Media Selection Logic

The Service Adaptation Function analyzes the properties of all alternatives, which are ordered according to the quality of semantic information, and tests if the Access Mechanism is capable of presenting one of the alternatives. Examples for properties are the type of the media element, its size, or number of colors. The user interface capabilities and additionally the user preferences are taken into account by the selection process. Accordingly, the most suitable media element is inserted by reference or by inline data into the target document. If no alternative is suitable, according conversion strategies are determined as depicted in Figure 3-53.

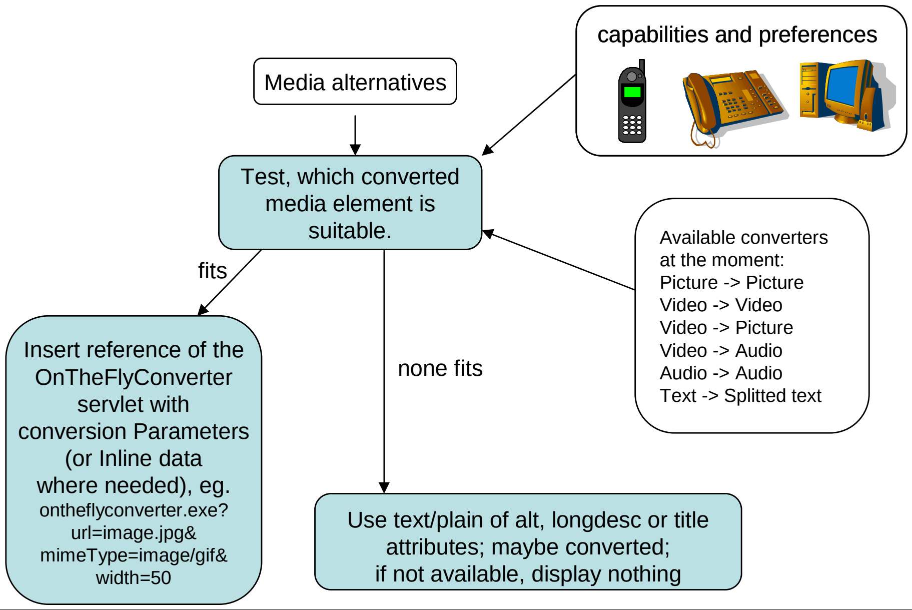

capabilities and preferences
Media alternatives
Test, which converted media element is suitable.
fits
none fits
Available converters at the moment:
Picture -> Picture
Video -> Video
Video -> Picture
Video -> Audio
Audio -> Audio
Text -> Splitted text
Insert reference of the OnTheFlyConverter servlet with conversion Parameters (or Inline data where needed), eg.
ontheflyconverter.exe?url=image.jpg&mimeType=image/gif&width=50
Use text/plain of alt, longdesc or title attributes; maybe converted; if not available, display nothing

Figure 3-53: Media Conversion Logic

The Service Adaptation Function evaluates the properties of all media elements, which can be provided by conversion in order to determine the most suitable media element. If a suitable element is found, the converted element, inline or a reference to the media converter with the according control attributes, is inserted into the document. If no suitable media element can be obtained by the available media converters, an alternative plain text representation, extracted from the meta-data describing the media element such as ‘abstract’ or ‘title’, should be inserted.

In order to reach the flexibility, necessary to support various types of user interfaces, the media converters should support the conversion of following media types and their different formats:

- Image
- Video
- Audio
- Speech
- Video & audio
- Text

This means, the Service Adaptation Function should dispose of an extensive pool of media converters, which can convert the media format (image:JPEG → image:GIF) as well as the media type (text → speech). It depends on the actual platform and the applications to be supported, which kinds of user interfaces shall be provided, what kinds of media type are used by them, and hence, which media converters have to be provided in the Service Adaptation Function.

Basically, as explained above, the Service Adaptation Function implements the I-centric User Interaction while representing a proxy gateway between the various supported user interface technologies and the service (reference points A and B in Figure 3-49). It hides the peculiarities of the different user interface technologies enabling the service to interact with the user without needing to support directly these technologies. In this sense, the Service Adaptation Function can be regarded as a kind of a middleware for user interaction, which enables the effortless development of services supporting them with powerful user interaction possibilities. The concept depicts an outsourcing of significant areas (such as adaptation, specific user technologies support, etc.)

3.5.6 Delivery Context

As described in the previous section, the Service Adaptation Function adapts the general user interface description, provided by the services in GUIML format, into specific user interfaces and vice versa. This process is controlled by considering all available and relevant data about the user, the environment, and the technologies used such as terminal and network. These data are described by the term *Delivery Context*.

The term Delivery Context was introduced in [W3CDIPRINC] to refer to the set of attributes that characterize the delivery environment. Among these attributes, the ones that are most relevant for achieving Device Independence are those that characterize the presentation capabilities of the Access Mechanism, the delivery capabilities of the network and the presentation preferences of the user. The Access Mechanism depicts the combination of hardware, including one or more devices and network connections, and software, including one or more user agents, which allows a user to perceive and interact with the service using one or more interaction modalities, e.g. sight, sound, keyboard, voice, etc.

The Delivery Context is one key component to realize I-centric User Interaction, because it requires the intelligent adaptation of the interaction to the particular situation, the user is in. This means, the Delivery Context plays a decisive role in the adaptation of the user interaction. Besides the capabilities of terminals and networks used in the interaction, the Delivery Context also contains user preferences and relevant ambient information as specified by following definition:

The *Delivery Context* comprises attributes, which characterize the capabilities of the Access Mechanism, the preferences of the user, and the relevant ambient information describing the user’s environment.

According to this definition, the Delivery Context contains the following information:

- Capabilities of the Access Mechanism, including
    - Network characteristics
    - Terminal capabilities
- User preferences
- Ambient information

The user preferences are provided by the Personalization model (section 3.3). The ambient information is provided by the Ambient Awareness model (section 3.4). In the following, the specification of the Delivery Context concentrates first on the capabilities of the Access Mechanism. However, the complete Delivery Context necessary to realize I-centric User Interaction should comprise all three kinds of information. The according language to describe the Delivery Context supports all these information. It depicts a kind of a basket, which is interpreted by the Service Adaptation Function and which is filled with data from the three defined models as shown in Figure 3-54.

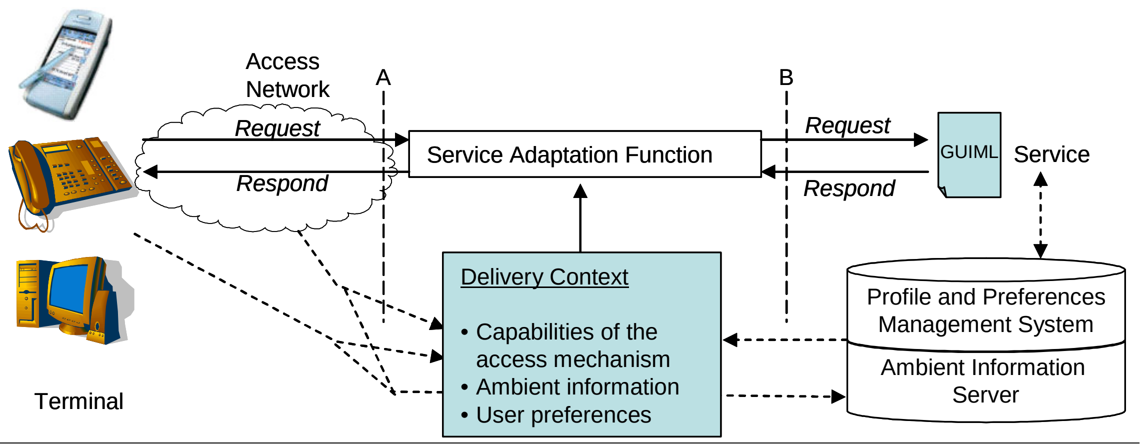

Access Network
A
B
Request
Respond
Service Adaptation Function
Request
Respond
GUIML
Service
Delivery Context
• Capabilities of the access mechanism
• Ambient information
• User preferences
Profile and Preferences Management System
Ambient Information Server
Terminal

Figure 3-54: Usage of the Delivery Context in Service Adaptation

Taking a more detailed look at the Access Mechanism, it can be recognized that the connection between the application on the user's terminal and the service consists of a number of elements. Each element has specific individual capabilities. Therefore, the capabilities of the complete Access Mechanism consist of the capabilities of all elements including terminal and network elements. On the one hand, proxy servers, used in an interaction session, can limit the transfer rate, but on the other hand, can provide additional conversion features.

If the terminal can only display a specific media type, but the proxy server is able to convert other media types to the one supported by the terminal, the capabilities of the complete Access Mechanism comprise the range of media types supported by the proxy server. In this way, the capabilities of the Access Mechanism have to be collected as depicted in Figure 3-55.

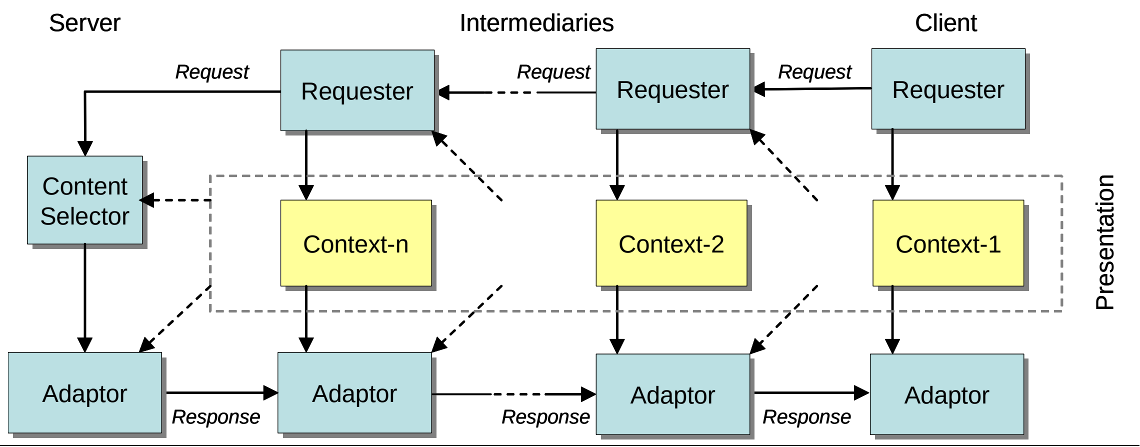


Figure 3-55: Delivery Context Chain

The initial request from the terminal is transferred to the service through the elements of the connection path, the intermediaries. The request is forwarded from element to element, which are therefore considered as requestors, too. The sequence of requestors provides additional Delivery Context information at different points in the request path from client to server. Similarly, a sequence of adapters can modify the response in the response path between the server and the client. The response can be modified based on any Delivery Context information available at that point in the response path. In some situations, the Delivery Context information can be available only to a restricted set of adapters. A requestor may block context information from being passed further along the request path, or make it available only to an associated adapter. For example, a phone can pass information about its location only to the corresponding gateway of the mobile network, which in turn does not necessarily provide this information to the target system.

There are a number of available concepts to obtain information about the terminal capabilities, which might be adequate for the limited tasks and application scenarios, but if it comes to advanced tasks and a broader range of applications, i.e. more data to be described by the Delivery Context, they are definitely not sufficient. One possibility is to extend existing approaches like WAP UAProf for example and to find workarounds for some of the problems, but then the solution would probably be difficult to use and to handle.

A requestor and adapter can act together as an element in the delivery path providing a specific part of the adaptation. The requestor may modify the request, including providing new context information, in such a way that the response can be adapted appropriately. For example, a transcoding proxy can offer to process a media type, which is not support in the original request. The Delivery Context is then to be extended to reflect the additional media type, likewise. When the response is received by the proxy, the additional media type can be adapted to one that is acceptable to the originator of the request. One example of this approach is the WAP gateway, which translates textual WML into a binary representation to be transmitted over air interface to the terminal.

In order to realize the adaptation to the Delivery Context, an XML-based description language for capability information was developed for the I-centric User Interaction system. This language is denoted as the *eXtensible Delivery Context Language* (XDCL)[30]. CC/PP [W3CCCPP] and UAProf [WAGUAPRSP] have served as the base for the design to ensure interoperability. Accordingly, the various building blocks of XDCL were developed corresponding to the structural elements of the existing standards.

To recall the structure of CC/PP with the underlying Resource Description Framework (RDF) [RDF], the major constructs are listed here:

- The concept of attributes is introduced in RDF as statements about resources, i.e. properties of a device.
- CC/PP defines components that group these attributes.
- Resolution rules in the vocabulary definition identify the behavior if an attribute occurs in a profile more than once.
- The CC/PP defaults resemble a special resolution rule and external defaults allow components that are stored externally to be included in a profile.
- CC/PP introduces the concept of ‘profile diffs’, which is used to convey changes in the profile in an efficient way.
- Another artifact from RDF is the distinction of three kinds of sets: sequences, bags, and alternatives.

XDCL applies XML and RDF. It defines a number of language elements to describe the Delivery Context. The elements reuse as much as possible the available components known from other concepts like CC/PP. The following sections introduce the individual elements of XDCL detailed.

3.5.6.1 Profiles

A Delivery Context is described by a *profile*. The profile contains all the data characterizing the Delivery Context. These data are specified as attributes of the profile. In that sense, the profile is a container that is created when a communication session starts, i.e. when the communication relationship between user and service is established. The profile is responsible for the management of the context delivery data. All components in the communication path (i.e. from user’s terminal, the network between, and the service platform) provide data describing the context,

[30] An initial approach of the XDCL was introduced in the diploma thesis [JASPER].

which have to be incorporated into the profile. In XDLC, a profile has basically the following structure:

```
<profile>
   [Attributes]
</profile>
```

The profile can have static and dynamic attributes. Its structure enables a flexible and efficient management as well as evaluation of the Delivery Context. The Service Adaptation Function and the service itself have access to the profile in order to adapt their behavior and their presentation accordingly.

3.5.6.2 Profile Attributes

RDF statements about resources are used in CC/PP to hold the actual attributes that describe the device capabilities and preferences. In essence, they are equivalent to name value pairs in this context. In CC/PP profiles, the subject of the RDF ‘subject, predicate, object’-triple can only be either a component or a default for a component and thus contains no essential information. The attribute element is introduced as the most basic elements of XDCL and is identified by its ‘name’ that has an associated ‘value’. For example, the following profile has two attributes: ‘screenwidth’ and ‘screenheight’:

```
<profile>
   <screenwidth>1024</screenwidth>
   <screenheight>768</screenheight>
</profile>
```

3.5.6.3 Components and Modules

Components in CC/PP are used to group similar attributes. An attribute can only belong to one component. Components have no parameters themselves. They are used to structure the information contained in a profile. Furthermore, components are important to ensure the extensibility in CC/PP as the definition of new components can be added in additional vocabularies.

Mark Butler expresses his view on components in “Some Questions and Answers on CC/PP and UAProf” [BUTLER1]:

> *“I think the confusion about components stems from the fact it is not obvious what purpose components serve. The CC/PP structure and vocabularies document recommends that attributes should really only be associated with one component: hence components don’t provide any additional information beyond attribute name. Furthermore, I observe that there has been disagreement in UAProf about which component is most appropriate for various attributes e.g. should ‘CcppAccept’ be in ‘SoftwarePlatform’ or ‘BrowserUA’? If we don’t have a concrete use case for why we need components, perhaps we need to exercise Occam’s razor and just dispose of them altogether.”*

In accordance with this, XDCL does not contain a component construct. Instead, the extensibility of a vocabulary is allowed by the use of XML namespaces. That means that a capabilities profile can be assembled from attributes defined in different vocabulary *modules*, i.e. identified by corresponding XML name spaces. The example below shows a profile that uses the vocabulary definition from a ‘screen-module’:

```
<profile xmlns:screen="http://www.example.org/screen-module#">
   <screen:screenwidth>1024</screen:screenwidth>
   <screen:screenheight>768</screen:screenheight>
</profile>
```

For example, possible modules can comprise a ‘core’ module defining all basic elements of the capabilities structure, a ‘screen’ module including the properties of the device’s screen, or a ‘WAP’ module, which is included if the device supports the WAP standard. To use modules instead of components is indeed be much more in line with several standards like XHTML, SVG[31], CSS, and SMIL, which all use the ‘modularization’ approach. Typical combinations of modules would then form ‘profiles’, for instance a ‘mobile phone profile’ or a ‘desktop web browser profile’. Section 7.3.5 contains the definition of some basic modules.

3.5.6.4 Resolution Rules and Defaults

A profile can be assembled by multiple entities on the transmission path from the terminal to the server, i.e. profile of terminal, network, gateways and proxies, etc. In CC/PP, this is ensured by the possibility to assemble multiple profiles into one, where the multiple entities are represented by RDF resources. XDCL offers this possibility, too. As there is no RDF used, the simple approach is just to append the attributes to the existing profile. This can result in a problem however: The same attribute could appear in a profile more than once as it is defined by different entities. In that case, a system processing the profile has to decide which value to take for the attribute. Appropriate resolution rules for those conflicts have to be defined.

Profile resolution means the evaluation of the profile data to reduce it to a common dominator. For example, if a terminal is able to play a video stream with up to 128k/bits, but the network transfer rate is restricted to 64k/bits, the resolution should determine the 64k/bits as the common dominator. This value has to be provided to the Service Adaptation Function in order to adapt the content accordingly.

The only resolution rule that is explicitly named in CC/PP covers the defaults for an attribute, which are always overwritten. UAProf defines the resolution rules ‘locked’, ‘override’, and ‘append’ for sets of values. The solution, proposed in this thesis, is that all attributes are described with ‘*quality value*’ between ‘0’ and ‘1’. This allows a much finer granularity. The resolution rule is simply that the attributes with the higher quality value is considered overwriting the same attributes with a lower quality value.

If two attributes have the same quality, the one that was inserted into the profile at last is assumed as having a higher relevance. Therefore, the order of the elements, which have been added to the profile, has to be preserved. If no quality value is specified in the profile, the default value defined for that attribute in the vocabulary module is taken. The UAProf ‘locked’ resolution rule would correspond to a quality value of ‘1’ and ‘override’ would be represented by a quality value of ‘0’.

The ‘transferrate’ attribute in the following example is resolved to the value ‘64’, because it has the highest quality:

```
<profile xmlns:net=”http://www.example.org/network-module#”>
   <net:transferrate q=”0.7”>128</net:transferrate>
   <net:transferrate q=”0.3”>512</net:transferrate>
   <net:transferrate q=”0.7”>64</net:transferrate>
</profile>
```

The CC/PP concept of ‘defaults’ can be emulated easily with this technique. Attribute values that shall be considered as ‘defaults’ have a quality value of ‘0’ and are therefore overwritten by any other attribute of the same type. In addition to this concept, CC/PP allows defaults to be external, i.e. stored in another profile, for example on a server of the manufacturer. In the process of profile resolution, these external defaults are loaded and assembled to one profile following the resolution rules. The corresponding concept in XDCL is named ‘*includes*’. It allows

[31] SVG: Scalable Vector Graphics format developed by Adobe as an extension to XML

referring to a local or external profile by an URI[32] and additionally declaring a quality value for the completely included segment. With this construct, it is possible to assemble a profile from many files and to assign some of them as external defaults by giving them a quality value of '0'.

The next example has two external include statements. The first one depicts an external default.

```
<profile xmlns:net="http://www.example.org/network-module#">
   <include q="0.0" uri="http://www.example.org/default-profile#" />
   <include q="0.7" uri="http://www.example.org/profile17#" />
   <net:transferrate q="0.9">64</net:transferrate>
</profile>
```

The first include statement (URI="http://www.example.org/default-profile#") refers to the following profile:

```
<profile   xmlns:net="http://www.example.org/net-module#"
           xmlns:screen="http://www.example.org/screen-module#" >
   <net:transferrate>512</net:transferrate>
   <screen:screenwidth>800</screen:screenwidth>
   <screen:screenheight>600</screen:screenheight>
   <screen:imagecapable>Yes</screen:imagecapable>
</profile>
```

This depicts the default profile, because quality value of the reference is '0'. This means, if no further information is available, which overwrites the profile data, these profile data have to be considered for the adaptation process. The second include statement refers to the following profile:

```
<profile xmlns:net="http://www.example.org/network-module#">
   <net:transferrate>128</net:transferrate>
</profile>
```

Resolving the include statements leads to the following complete profile, which is the base for the resolution process:

```
<profile   xmlns:net="http://www.example.org/net-module#"
           xmlns:screen="http://www.example.org/screen-module#" >
   <net:transferrate q="0.0">512</net:transferrate>
   <screen:screenwidth q="0.0">800</screen:screenwidth>
   <screen:screenheight q="0.0">600</screen:screenheight>
   <screen:imagecapable q="0.0">Yes</screen:imagecapable>
   <net:transferrate q="0.7">128</net:transferrate>
   <net:transferrate q="0.9">64</net:transferrate>
</profile>
```

It will be very common that profiles are constructed as a list of includes with URIs to profiles stored in a profile repository. Apart from dynamic attributes, client devices and intermediary proxies for example normally do not alter values in a profile but only submit their static profile. Because a profile just consisting of a number of URIs to external profiles is naturally small, this concept is quite efficient and do not stress the complete system with too much management overhead. This concept depicts a balanced compromise to describe the capabilities of the Delivery Context.

[32] URI: Universal Resource Identifier - a compact string of characters for identifying an abstract or physical resource. [RFC2396]

3.5.6.5 Profile Diffs

Another concept to reduce the amount of data to be transferred is the introduction of *profile diffs*, originally introduced with CC/PP. If some modifications to a profile – either to the defaults or to the actual profile – are to be made, only the specific values that have changed have to be transmitted again. XDCL provides the same mechanism. If a Delivery Context is already known, for example in an existing Access Session, a *profile diff* that is received as part of a request is appended to the last profile eventually replacing older values according to the given quality values.

A profile with too many 'diffs' raises a potential problem: normally profile diffs are simply appended to the profile keeping as much information as possible until the final profile resolution is performed to calculate the current attribute values. In environments with a very dynamic Delivery Context, for example containing the location of a moving user, the profile will grow extremely large and probably will be unusable in a very short time. At least, such permanently growing profiles will affect the system performance negatively. Therefore, the implementation has to decide carefully, which data are kept in the profile and which repeated updates from the same source should be discarded keeping the most up to date value only. In the worst-case scenario, only the latest update is included in the profile overwriting the previous updates. In the following example, a complete profile was negotiated at the beginning of a Service Session:

```
<profile xmlns:net="http://www.example.org/network-module#">
   <include q="0.0" uri="http://www.example.org/default-profile#"/>
   <include q="0.7" uri="http://www.example.org/profile17#"/>
   <net:transferrate q="0.9">64</net:transferrate>
</profile>
```

The 'transferrate' attribute depends on certain conditions and is therefore dynamic. In order to provide the current value to the Service Adaptation Function without transmitting the complete profile again, the terminal sends only updates using a profile diff as follows:

```
<profilediff xmlns:net="http://www.example.org/network-module#">
   <net:transferrate q="0.9">48</net:transferrate>
</profilediff>
```

The complete profile, to be processed by the profile resolution and then to be considered by the Service Adaptation Function, looks as follows:

```
<profile xmlns:screen="http://www.example.org/screen-module#"
   xmlns:net="http://www.example.org/network-module#" >
   <net:transferrate q="0.0">512</net:transferrate>
   <screen:screenwidth q="0.0">800</screen:screenwidth>
   <screen:screenheight q="0.0">600</screen:screenheight>
   <screen:imagecapable q="0.0">Yes</screen:imagecapable>
   <net:transferrate q="0.7">128</net:transferrate>
   <net:transferrate q="0.9">48</net:transferrate>
</profile>
```

3.5.6.6 Sequences, Bags, and Alternatives

Some attributes can have more than one value. In the examples above, the attributes define only one specific value, e.g. 'screenwidth'='800'. This actually depicts an upper or lower limit merely. The real meaning is not necessarily clear. In fact, they mean usually a range or some possibilities. Therefore, the attributes should be described more precisely by specifying values that are more concrete. In RDF, three different relevant data types are defined: sequences, bags, and alternatives. CC/PP and UAProf adopted these constructs. However, the declaration of a list of values using a bag, a sequence, or an alternative in the profile does not contain much more semantic information compared to the multiple specifications of the same attribute but with different values. The semantics for attributes should be defined in the vocabulary definition alongside its data type.

Another possibility, which is used in XDCL, is to specify the profile attributes with sub-elements, i.e. child elements. Each child element depicts one alternative. The main profile attribute represents the list itself and the child elements represent the items in the list. This approach opens the opportunity to declare more complex data types like additional parameters to list elements. As list elements also have a quality value, it is easily possible to state preferences between the alternatives in the list. The following example shows a profile containing a list of different supported transfer rates, each marked with a corresponding quality value:

```
<profile xmlns:net=”http://www.example.org/network-module#”>
   <net:transferratelist>
      <net:transferrate q=”0.9”>128</net:transferrate>
      <net:transferrate q=”0.4”>64</net:transferrate>
   </net:transferratelist>
</profile>
```

3.5.6.7 Sources

Besides the structuring of profile attributes into modules and list, it is also necessary to track the origin of the attribute values. Multiple entities, e.g. terminals or gateways, could specify a value for the same attribute and the priorities to resolve those conflicts, i.e. to select the most relevant entry, depend on the kind of entity. For a reasonable evaluation of the data contained in a Delivery Context, it is meaningful to know the origin of the data and to know information about the origin. It is not necessarily sufficient to evaluate context data just based on their values, but also on their origin. The primary goal is to describe attribute as good and detailed as possible. However, this has to be specified in the common module definition or language profile. Assuming that such static profile definitions only cover a subset of data, which are sufficient at a certain moment in time, a dynamic approach is required, likewise. The dynamic approach can be achieved by identifying the source of information and by providing additional information. This approach is applied in XDCL.

Consequently, in XDCL all profile attributes can be described with a ‘source’ parameter. The value of this parameter shall uniquely identify the entity, which has provided the data. This source identifier can be used for the interpretation of the profile data. The definition of the identification scheme is not in the scope of XDCL, but has to be defined in the actual implementation. The following example describes a profile with location data derived from a GPS sensor with source identifier ‘GPS-0815’, from a GSM network Location Server with identifier ‘GSM-0175’, and from an active badge sensor with identifier ‘BADGE6006’:

```
<profile xmlns:loc=”http://www.example.org/location-module#”>
   <loc:longitude source=”GPS-0815”>13.18831E</loc:longitude>
   <loc:latitude source=”GPS-0815”>52.31555N</loc:latitude>
   <loc:longitude source=”GSM-0175”>13,15E</loc:longitude>
   <loc:latitude source=”GSM-0175”>52,34N</loc:latitude>
   <loc:longitude source=”BADGE6006”>13,188E</loc:longitude>
   <loc:latitude source=”BADGE6006”>52,315N</loc:latitude>
</profile>
```

Because there is no detailed information about the quality of the data in the profile, the additional data about the sources can be considered for the profile evaluation. The additional data should then contain information about the resolution and accuracy of the different techniques.

3.5.6.8 Device Classes

For example in WAP UAProf, profiles are defined by an extensive list of device attributes giving a very detailed description of the properties of the device. On the one hand, this is advantageous for adaptation process because a very good representation for a specific device can be achieves since of the high number of parameters to use. On the other hand, this process is complicated and can be time-consuming depending on the available computational resources. Especially if there is no automated adaptation, the creation and selection of predefined presentations, in the form of XSL style sheets for example, according to the huge amount of attributes is costly.

An alternative or additional approach to the construction of a detailed vocabulary is the definition of *device classes*. A device class groups all devices with similar properties, for example all small-screen low-bandwidth colorless phones. It is simpler to specify a suitable adaptation algorithms for individual classes than for each detailed vocabulary considering every possible single device attribute in it. The adaptation function and the services can then select the appropriate algorithm according to the given device class.

Because of the coarse granularity of this separation, this approach should be used only for rather basic and simple adaptation. There has to be a fixed set of rules for each device class that define the characteristics of devices, which belong to particular group. For example, a group ‘small-screen’ could be defined as: ‘The width of the screen has to be smaller than 200 pixels and the height has to be smaller than 150 pixels’. A device will be considered as belonging to a class if it matches to all the characteristics of that class.

The individual device classes have to be defined in a way that a device can at least fit into one group – if it does not, a new class must be created for it. Of course, this depends on the level of abstraction regarding the attributes, which are considered in the definition of a device class. As more attributes are chosen, the more device classes have to be defined. In order to profit from this simplified approach, well-balanced device classes have to be defined. This approach is partially applied already today in systems, which adapt the content for example based on the user agent string and additional HTTP header information supplied by the browser on the user’s terminal. In this way, today’s WWW and WAP servers prepare the content specifically for the different WAP terminals and WWW browsers.

As a next step, hierarchies of device classes are introduced to differentiate them further and to make the adaptation process easier. Several device classes are grouped in a collection that describes a similarity of all devices, which are included in the respective group; e.g. ‘small screen’ containing the ‘PDA’ and ‘Phone’ device classes and ‘big screen’ containing ‘Laptop’ and ‘Desktop computer’. Figure 3-56 presents an exemplary structure of device classes. The content can now be adapted according to that higher-level, i.e. more abstract, class, likewise. This procedure can reduce the number of necessary adaptation algorithms further. For instance, it may be sufficient to adapt the content to size of screen (small or big) without considering the further characteristics of the terminal, e.g. PDA, phone, laptop, or desktop computer. The distinction is limited and the classes have to be defined thoroughly. For example, if devices are divided by screen size, the device classes cannot make any statement about the available bandwidth.

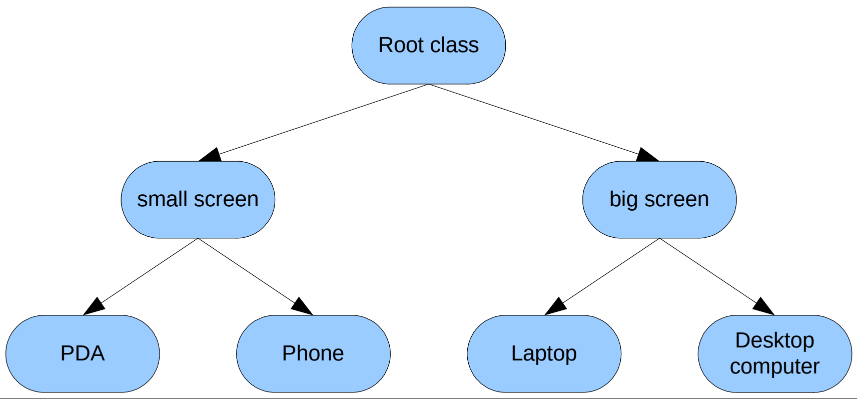


Figure 3-56: Exemplary Device Class Hierarchy

The approach of using device classes means a simplification of using detailed and well-defined device capabilities as proposed in the previous sections. It does not depict a compensating alternative, but is an additional approach for specific kinds of applications, which do not have high requirements on the Service Adaptation. In this sense, it can be understood as an additional concept hiding the complexity of the general adaptation mechanism. The implementation can provide both the detailed attributes of the Delivery Context and the device class derived from these attributes to the Service Adaptation Function and to the services as shown in the following example:

```
<profile   xmlns:screen="http://www.example.org/screen-module#"
           xmlns:general="http://www.example.org/general-module#">
   <screen:screenwidth>1024</screen:screenwidth>
   <screen:screenheight>768</screen:screenheight>
   <general:deviceclasses>/big screen/Laptop</ general:deviceclasses>
</profile>
```

3.5.6.9 Capabilities classes

It is obvious that the introduction of device classes can simplify the work of service authors or rather the equipment manufacturers. However, depending on the level of abstraction there can be a huge amount of different device classes. Implicitly contained in the definition of device classes is the introduction of some higher-level attributes like 'small-screen' or 'low-bandwidth'. In [BUTLER2], Butler calls these clusters *capabilities classes*. In this approach, devices are not assigned to specific device classes, but they are assigned to multiple capabilities classes. A capability class describes a certain capability, e.g. the screen size, bandwidth, etc. In contrast to define super classes for the devices, i.e. the device class approach as described in section 3.5.6.8, the devices are linked to multiple corresponding capability classes. Accordingly, the device is described by a selection of the capability classes that suits to the capabilities provided by the device. Figure 3-57 shows a simple example of the assignment of capability classes. In a real scenario, the number of available (identified) capability classes is clearly higher.

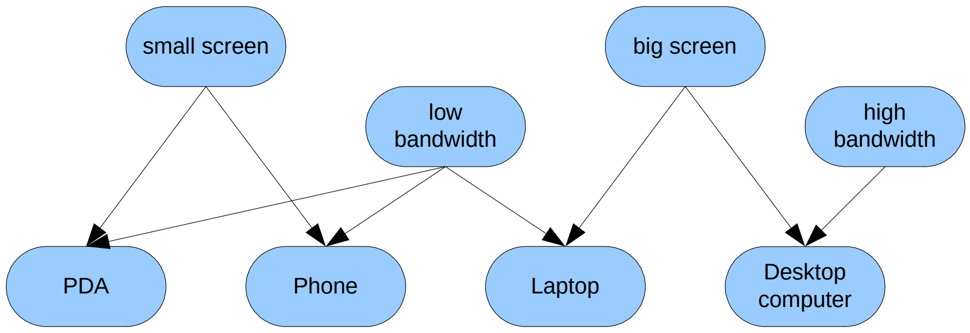


Figure 3-57: Exemplary Capability Class Hierarchy

The assignment of devices to specific capabilities classes is appropriate to the Service Adaptation Function. The adaptation process can be optimized, because it can easily access the relevant data, which are important to the process. For example, if the service provides images, which are to be presented to the user, the adaptation function can easily check the Delivery Context, whether the device has a graphical output capability, i.e. a screen, and, if so, the size of the screen in order to prepare the images for the presentation.

The advantage of capabilities classes compared to device classes is the much finer granularity and that there is no need to introduce another class for every possible combination of applicable capabilities classes. Therefore, this approach is a compromise between the very coarse adaptation according to device classes and the adaptation to the possibly huge amount of attributes in the Delivery Context.

In XDCL, a slightly different variation of both presented approaches is used. Instead of just relying solely on device classes or capabilities classes as input parameters to the transformation process, both specifications are adopted in the Delivery Context description as additions to the direct descriptions of capabilities. Thus, the transformation process or the content author can decide freely which approach to take and which granularity of attributes to choose.

In the process of assigning devices to capability classes, there are two options. The terminals can be assigned to groups by the creator of the capabilities profile, i.e. the manufacturer of the device, or according to the rules as explained above. In the second case, there are two further options. A device can be assigned to a class either at the time when the profile is processed or when it is initially created. This is implementation specific and has no impact on the vocabulary structure, as the structure will allow the declaration of device and capabilities classes in any case.

3.5.6.10 Vocabulary of XDCL

While XDCL offers the syntax for the description and the transmission of Delivery Context profiles, a vocabulary is needed to define the semantics of what is included in the specific profiles. This means, a vocabulary defines the possible attributes and their meaning. The usage of a vocabulary is the base for a common understanding of the data, which should be contained in the Delivery Context.

A widely adopted standard so far is the WAP UAProf vocabulary [WAGUAPRSP]. As it was defined by the Open Mobile Alliance (OMA – formerly WAP Forum) [WWWOMA], it concentrates on the description of mobile phone devices supporting WAP. However, there are more types of devices and communication services than WAP. In [BUTLER], this is described in the following way:

"The WAP User Agent Profile (UAPROF) is a standard being developed by the WAP Forum. It is intended that future WAP devices will use it to communicate their capabilities to a server. It is a CC/PP application so it addresses the first problem highlighted in the previous section i.e. provides a standard vocabulary for WAP clients to communicate their capabilities to servers. However it is not ideal as it would be preferable to have a single vocabulary for all web clients rather than just WAP devices."

In order to support real Device Independence, the vocabulary should not be restricted to a specific technology and specific kinds of devices. In the contrary, it should support all data relevant to describe a Delivery Context according to the general requirements of I-centric User Interaction. It has to be considered carefully, which data should be included in the profile. Beside the Service Adaptation Function, also the services themselves as well as possibly further components in the I-centric User Interaction system use these data in order to adapt to the context. In [W3CDCODI], the selection of the relevant data is described as follows:

"Among these attributes, the ones that are most relevant for achieving Device Independence are those that characterize the presentation capabilities of the Access Mechanism, the delivery capabilities of the network and the presentation preferences of the user."

As the focus of the I-centric User Interaction approach concentrates on personalization, ambient awareness, and Device Independence, the XDCL vocabulary has to support attributes that describe the input or output capabilities, user preferences, and ambient information. Attributes describing the preference data are provided by the Personalization model (section 3.3), and the attributes describing as ambient information are provided by the Ambient Awareness model (section 3.4). The vocabulary proposed in this thesis does not depict a definite solution – it shows an application of XDCL and demonstrates how a reasonable vocabulary can be constructed. The following paragraphs propose some exemplary module specifications to support the Device Independence. The standardization of capabilities should remain the duty of device manufacturers and specialized working groups.

A first step in the creation of a vocabulary could be the translation of UAProf into XDCL to fill the vocabulary with a widely adopted base set of attributes. This would be reasonable, as the new vocabulary would be compatible to UAProf. In the next step, some other common vocabularies like the CC/PP core and proxy vocabularies could be integrated. Then, this collection of attributes has to be analyzed for redundancies, inconsistencies, and completeness. The vocabulary will be consolidated and missing attributes for additional device classes will be inserted. A sensible division into modules and possible targets shall be found at the same time.

The set of modules in the XDLC vocabulary should rather resemble the very fine granularity of the choice of modules in XHTML than the quite rough division into the UAProf components. It is not always clear where to draw the line between two modules, like in the following example of a mobile phone: It is obvious that the properties of the screen will belong together in one group if we look at the phone as a WAP device. However, if we consider the phone as an SMS client, capabilities like screen size in pixels or color capabilities are irrelevant because a SMS is text-only. This use case shows that the distinction has to be made between properties that are important to adapting text and properties that apply to graphics. There needs to be a 'text' component and a 'graphics' component instead of just a 'display' component or – even more general – a 'hardware' component.

If the modules are defined with the right granularity and in sensible combinations, the requirements to describe a wide variety of devices by the vocabulary and to encourage re-use modules are met. In the sense of re-use, redundancies between the modules shall be avoided. In UAProf, for example, a number of components have similar attributes, only distinguished by a prefix to determine the component, as mentioned above in section 3.5.6.3. For instance, the three attributes 'Name', 'Vendor', and 'Version' appear in the UAProf vocabulary in the 'Hardware', 'BrowserUA', and 'WAP' component – with the respective prefix and with slightly altered names. In the sense of modularity, this should be avoided. Instead, an 'Info' module is introduced in XDCL providing 'Name', 'Vendor', and 'Version' attributes. The 'Info' module can then be used together with other modules with a unified syntax and semantic.

Following this idea, a set of initial modules for XDL was defined, which is presented in Table 3-9. These modules form the basic XDCL vocabulary and provide a suitable base to re-use UAProf attributes. In this way, the compatibility to existing approaches is ensured.

Module	***Description***
General	general properties that do not belong to any other category, for example the capability classes or the user agent string for the device
Info	'Name', 'Vendor', 'Version' and other general meta data for an entity
Hardware	properties of the hardware like kind of CPU and memory size
Content	properties that apply to the general content like the accepted content types, languages, and message sizes
Text	the attributes for text display like how many characters can be displayed on the screen
Input	the input modalities for the device, besides keyboards including graphical or audio input
HTML	information about an HTML browser, mainly about supported language constructs like XHTML modules or JavaScript versions
Graphics	the display capabilities for graphics – in contrast to a text display – like screen or window sizes, resolution, etc
Sound	the sound output capabilities, like supported sound formats and qualities
Network	general network capabilities and Quality of Service (QoS) parameters including available bandwidth etc
Bluetooth	Bluetooth properties like supported Bluetooth profiles
WAP	attributes specific to the transmission via the WAP standard
Push	general capabilities that are used for 'push' scenarios (mainly WAP push), which are not part of the WAP or MMS modules

Module	*Description*
MMS	attributes related to multimedia messaging service
Java	information about the Java VM and other Java specific settings
MExE	capabilities of the 'Mobile Execution Environment'
Video	information about supported video codecs, frame rates and progressive scan or interlaced
Location	the position of the device – retrieved from a GPS receiver for example
User Preferences	Normally, user preferences are stored within the other modules by overwriting the default capabilities. The preference for having no sound output for example is expressed by overwriting the default 'sound output' capability in the 'Sound' module with 'No'. A preferred language is stated by changing the 'acceptable languages' value in the 'Content' module. However, some preferences cannot be found in other modules, especially preferences unique to some service application. Services would have to define their own vocabulary modules in this case.

Table 3-9: XDCL Vocabulary Modules

This list of potential modules is subject for extensions, but contains already some relevant modules in order to describe Delivery Context data as used in real systems. It also serves as a pattern for defining furthers modules to extend and to refine the vocabulary according to the respective requirements of the envisioned applications. The individual modules, i.e. the attributes contained in each module, are listed more detailed in the appendix section 7.3.5.

The *user preferences* and the *location* modules are examples how data from the other models (Personalization model: section 3.3 and Ambient Awareness model: section 3.4) can be included in the Delivery Context. The models specify the concrete data (modules) to describe the according context information, see corresponding sections for details.

3.5.7 Conclusion

The mode for Generic User Interaction defines a language, denoted as the Generic User Interaction Markup Language (GUIML) to describe user interfaces in a device-independent way. This means, such user interfaces are not specific to any certain user interface technology. Instead, they can be transformed automatically to the respective representation as required by the particular communication service. This mechanism bases on the separation of the user interface and the service logic. Following this model, service developers can concentrate their effort on the service logic providing only a generic user interface. They do not have to implement the support for specific user interface technologies, which saves development costs and provides flexibility to access a service by any kind of user interface technology.

The transformation of the generic user interface to the technology specific representation is executed by the Service Adaptation Function. This function maintains suitable transformation rules for all supported user interface technologies. It has to consider the capabilities of the Access Mechanism consisting of terminal and access network. The Delivery Context describes these capabilities and controls in this way the concrete transformation process. Additionally, the Delivery Context contains user preferences and relevant ambient information, which should be considered in the transformation. The presented approach defines an open and flexible way to specify the Delivery Context by using the eXtensible Delivery Context Language (XDCL).

In this way, the user interaction is device independent, personalized, and ambient aware as required by the objectives of I-centric User Interaction. The information required for the personalized and ambient aware appearance is to be provided by the Personalization model and Ambient Awareness model.

3.6 Summary

This chapter introduced an approach to realize I-centric User Interaction. Based on the general vision and the more detailed objectives, chapter 2 has identified three individual areas of concern. These areas cover the personalization, ambient awareness, and device independence. Accordingly, chapter 3 presented three models respectively. These individual models define necessary concepts to provide suitable solutions for each area of concern.

Based on the three models for Personalization, for Ambient Awareness, and for Generic User Interaction, the approach to realize I-centric User Interaction defines a Service Adaptation Framework. This framework provides the necessary functional elements to implement the developed concepts and interfaces. Whereas the specification of the different models enables a separated realization of each, the proposed Service Adaptation Framework depicts a coherent realization of all three models. In this way, the objectives and principles of I-centric User Interaction can be realized. The Service Adaptation Framework depicts a platform for I-centric User Interaction providing the necessary functions and interfaces.

The framework specifies a transformation pipeline, which is executed by the Service Adaptation Function and which considers the relevant data obtained from the Delivery Context. This pipeline consists of a several steps necessary to transform the generic user interface into a user interface technology specific representation. Furthermore, the framework defines an interface, through which services can access the Delivery Context data, i.e. preference settings and ambient information, to behave personalized and ambient aware to the user.

The Personalization model defines an information model to structure and to describe user preferences. The concrete settings are defined as attributes consisting of name value pairs. There are different categories of preferences: the general user preferences and the service specific user preferences. Whereas the general user preferences contain common settings, relevant to all services, the service specific user preferences only contain the settings for a specific type of services. The concrete meaning of individual service preferences is known by the respective service only.

The user preferences are grouped into User Profiles, which can have a Selection Context. In this way, the user can specify the service behavior for different situations, which are described by the Selection Context. During the service usage, the Selection Contexts of all User Profiles are evaluated to determine the currently effective User Profile, whose preference settings are used accordingly. Beside this implicit activation, the user can also explicitly select a User Profile to be used.

Based on the evaluation of the relevant use cases, the Personalization model provides a specification of two interfaces for the internal and external access to the Preferences Management System. This separation reflects the need to ensure the privacy of personal data and to prevent any kind of misusage. On the one hand, the preferences settings are provided to the services directly. On the other hand, the preferences settings are provided to the Service Adaptation Function, which considers the user’s preferences in the realization of the user interaction.

The Ambient Awareness model specifies concepts and mechanisms to gather and to manage ambient information. It supports sensor networks as sources of ambient information, but also the Service Adaptation Function, which can derive ambient information from the Access Mechanism, i.e. from the terminal or from the user’s behavior. Furthermore, the model specifies a concept to interpret and revaluate raw data. The data, provided by sensor devices, are not necessarily sufficient for a reasonable adaptation. According interpreter components can revaluate such raw data to provide a more reasonable meaning. This depicts an open and flexible approach, which is not limited to a certain subset of available sensor technologies.

Furthermore, the Ambient Awareness model describes how ambient information can be used in an I-centric User Interaction system. Fundamentally, the ambient information is to be included in the Delivery Context. According to available location information, the ambient information storage can be searched for relevant data describing the respective environment, in which the user interaction takes place.

The model for Generic User Interaction proposes a language, denoted as the Generic User Interaction Markup Language (GUIML), to describe the user interaction in a technology independent way. This means that service developers do not have to implement a support for specific user interaction technologies, but can provide the service's user interface in a generic, technology independent description. The Service Adaptation Function transforms the generic representation of the user interface into the specific representation as required by the Access Mechanism.

On the one hand, the transformation includes the translation of the language, i.e. mapping of GUIML to a specific user interaction technology, such as WWW or WAP. On the other hand, the transformation considers the particular capabilities of the heterogeneous Access Mechanism consisting of the capabilities of the terminal and the capabilities of the network. In this way, the user interface is adapted to the Access Mechanism to support different kinds of terminals and different kinds of access network.

The adaptation of the user interface consists of the dialog adaptation and the content adaptation. Both aspects have to be solved individually. The dialog of a user interaction is realized in the various user interface technologies in different ways. This is because of their different nature and the different human senses, on which they are focused. Because of the heterogeneity of terminals and of networks, the content has to be adapted to the individual capabilities, likewise. The content adaptation comprises content selection, content splitting, and media conversion strategies.

The capabilities of terminals and networks, required for the adaptation, are described by the Delivery Context. This construct contains all data necessary for the adaptation process to realize device independent, personalized, and ambient aware user interaction. The preference data and the ambient information are provided by the Personalization model and by the Ambient Awareness model respectively. Accordingly, an open and flexible language to describe the Delivery Context – the eXtensible Delivery Context Language (XDCL) – is specified.

The presented approach to realize I-centric User Interaction contains all necessary concepts to fulfill the requirements and objectives of I-centric User Interaction. For evaluation and demonstration purposes, the individual concepts, i.e. the complete Service Adaptation Framework, were implemented prototypically. This realization is introduced in chapter 4.

CHAPTER 4 Realization

This chapter introduces a practical realization of an I-centric User Interaction system. This system implements the introduced Service Adaptation Framework supporting the different models as described in chapter three. Accordingly, this system supports the device independent user interaction as well as the personalized and ambient aware appearance of the services. Furthermore, some exemplary services using the I-centric User Interaction system for the interaction with the user are introduced to illustrate the usage of this system.

4.1 Introduction

The individual models to realize I-centric User Interaction as described in chapter 3 were developed in the course of several research projects. These research activities were accompanied with a practical development of a portal platform to implement and to validate the developed concepts. Accordingly, a number of exemplarily services demonstrating the full potential of the I-centric User Interaction approach were implemented and demonstrated. The experiences from the practical work have influenced the individual development steps and have facilitated necessary refinements of the concepts.

This chapter introduces the practical realization of the proposed I-centric User Interaction approach. This realization is denoted as the *I-centric User Interaction Portal*. It implements the models and concepts introduced in chapter 3. This realization depicts a proof of concept and demonstrates the potential of the developed approach. A refined architecture was derived from the introduced models and the according specifications describe concrete implementation guidelines.

For the actual implementation work, relevant target platforms were evaluated first. Based on the evaluation, the most suitable implementation and execution platforms were selected for the individual components. Special attention in the selection of technologies and the determination of the respective specifications was paid to the demonstration and evaluation of the individual models. Practical issues, such as runtime behavior and commercial applicability, were considered secondary only.

The following sections introduce the general architecture and describe the realization of the individual components accordingly.

4.2 Overview

The approach to realize I-centric User Interaction specifies three models providing suitable solutions for the different aspects:

- A model for Personalization
- A model for Ambient Awareness
- A model for Generic User Interaction

Furthermore, the approach includes a Service Adaptation Framework describing the functional elements and their interworking on a conceptual level. This framework, as introduced in section 3.2, depicts therefore a Platform Independent Model[33] without any consideration of specific development and execution platforms. During the development of the general I-centric User Interaction concept and of the individual models belonging to this concept, a corresponding implementation of a prototype was pursued.

Figure 4-1 shows a general overview of the developed I-centric User Interaction portal. This portal implements the individual models to realize I-centric User Interaction and provides respective interface to the services and to the access networks.

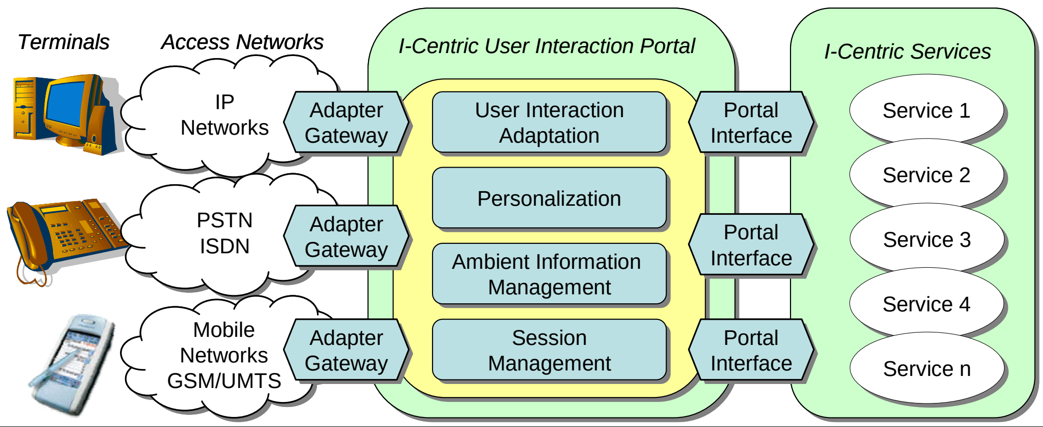


Figure 4-1: I-Centric User Interaction Portal Overview

The portal platform is connected via the Adapter Gateways to the different kinds of access networks. These gateways implement the specific protocols of the respective Access Mechanism[34] and provide all necessary functions to manage the connection to the user's terminal. According to the characteristics of the different supported Access Mechanisms, the Adapter Gateways support incoming and outgoing scenarios as well as synchronous and asynchronous communication.

For the prototype of the I-centric User Interaction portal, a number of different relevant communication services were selected and according Adapter Gateways were implemented (see section 4.5). Figure 4-2 shows an overview of the supported communication services.

[33] The concepts of Platform Independent Model and Platform Specific Model were introduced in OMG's Model Driven Architecture approach. See [OMGMDA].

[34] See section 2.5.4 for an explanation of the term Access Mechanism.

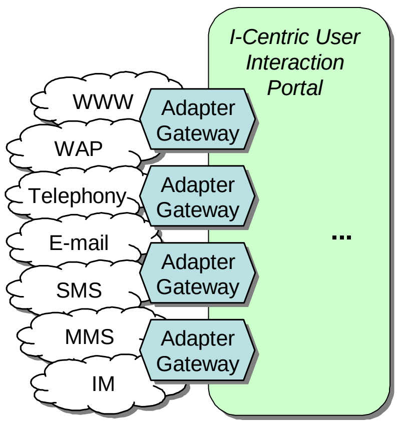


Figure 4-2: Communication Services supported by the Portal

Because of their different nature and used technologies, these selected communication services depict suitable and relevant examples of available user interaction technologies. They include synchronous and asynchronous communication paradigms as well as single-media and multi-media services. Table 4-1 shows an overview of the selected communication services supported by the I-centric User Interaction portal and describes the individual relevant characteristics.

Communication Service	***Characteristics***	***Comments***
World-Wide-Web (WWW)	Multimedia content User-driven access	Pleasant presentation possibilities on common computers, increasingly supported on small screen devices such as PDAs
Wireless Application Protocol (WAP)	Limit content (text and small pictures), restricted size User-driven and service-driven access (WAP-Push)	Huge number of terminals (i.e. mobile phones) and therefore increasing relevance expected
Telephony	Speech-based communication Synchronous	Because of technological restrictions not very comfortable and therefore not widely used user interface But, very relevant for particular environments such as cars
E-mail	Usually plain text, but in HTML formatted e-mails also multimedia content Asynchronous	Widely used today Relies on Internet access and capable devices (although mobile phone supports increasingly the receiving and sending of e-mails, too)
Short Message Service (SMS)	Plain text, limited size (n*160 characters) Asynchronous	Every GSM-based mobile phone supports SMS Regarded as killer application in GSM Already used today for delivery of news tickers and location-based services
Multimedia Message Service (MMS)	Multimedia content Asynchronous	Successor of SMS enhanced with multimedia support and extended functionality
Instant Messaging (IM)	Primarily plain text, but also additional services such as Voice-over-IP telephony and content exchange	Supports preferences and availability information Used today primarily for user to user and user to groups communication

Table 4-1: Characteristics of the selected Communication Services

As depicted in Figure 4-3, the portal platform provides a suitable interface to the services. This interface is used to receive the user interface to be presented in an adapted form on the user's terminal and to return according feedbacks from the user. Additionally, the interface provides the Delivery Context[35] data to the services necessary to enable a personalized and ambient aware behavior.

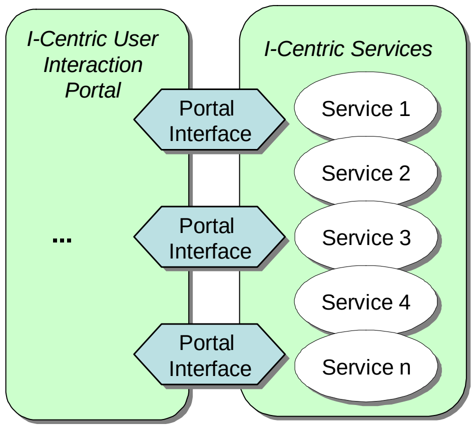


Figure 4-3: Service Access to the Portal

This means that the Portal Interface needs to exchange GUIML documents as specified in the Generic User Interaction model and provides access to the Delivery Context Handler as specified in the Service Adaptation Framework for I-centric User Interaction. The interface depicts an external reference point with regard to the distribution, but also with regard to the provider perspective. The services usually are realized and operated by service provider different to the portal provider. This requires a careful functional and technical design of the portal interface. On the one hand, the external domains of the service providers cannot be regarded as being reliable and secure. A service, which behaves erroneous or incorrect, must not influence the function of the portal at all. In addition, the transmissions in the network connection between portal and service provider, which can be the public Internet, have to be secured in order to ensure the privacy and data integrity.

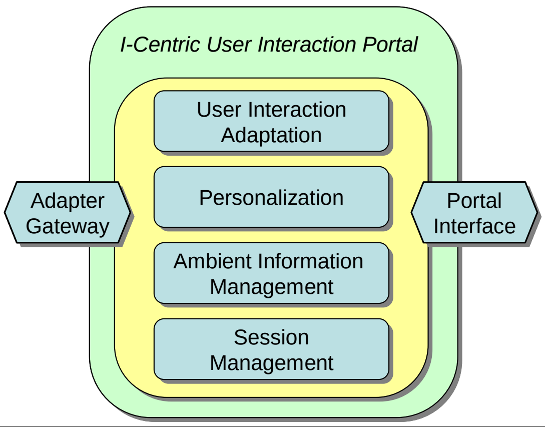

I-Centric User Interaction Portal
User Interaction Adaptation
Personalization
Ambient Information Management
Session Management
Adapter Gateway
Portal Interface

Figure 4-4: The Core of the I-Centric User Interaction Portal

As shown Figure 4-4, the implementation of the different models for I-centric User Interaction is distributed over a number of components, which represent the core of the portal platform. These core components depict the main functional areas.

[35] See section 3.5.6 for detailed explanation.

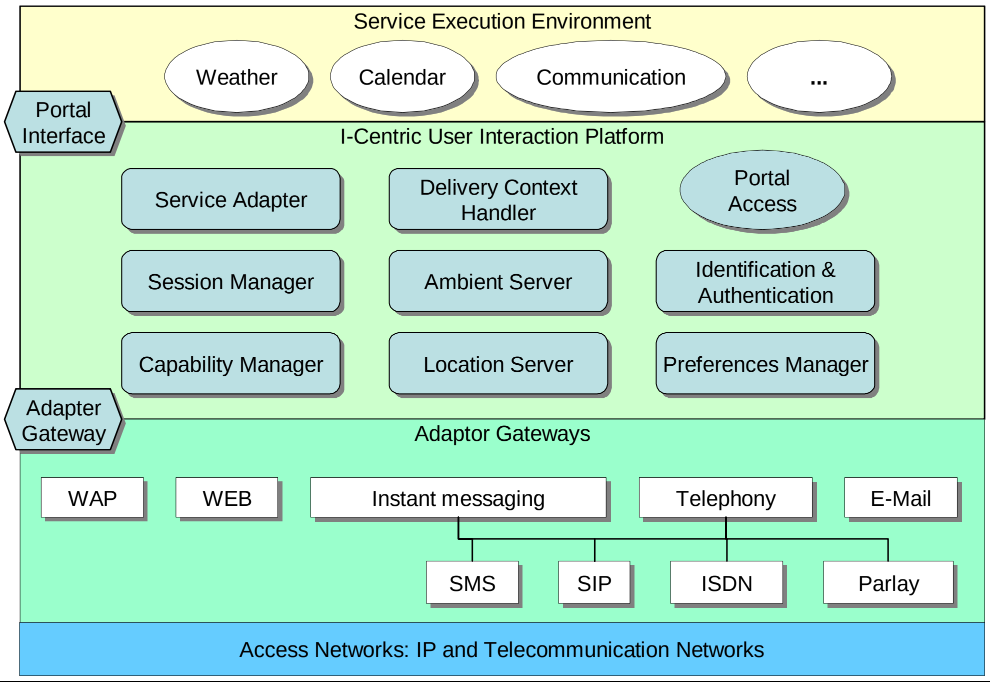


Figure 4-5: The Architecture of the I-Centric User Interaction Portal

Figure 4-5 presents the general architecture of the I-centric User Interaction portal. The services interact with the portal through the portal interface. The individual components in the I-centric User Interaction platform realize the three models for I-centric User Interaction. The actual communication with the user’s terminal is realized by the Adapter Gateways, which implement the protocols and concepts of the access networks.

4.3 Technology Viewpoint

The implementation of the I-centric User Interaction portal applies several technologies to implement the individual components. Whereas the complete Service Adaptation Function is realized as a transformation pipeline in the Cocoon XML publishing framework [WWWCOCOON, LANGHAM], other core components are realized as servlets in a WWW application server or as Enterprise Java Beans [SUNEJB] in an EJB application server. All components are implemented in the Java programming language, apart from the transformers of the Service Adaptation Function, which are realized as XSL style sheets [W3CXSLT].

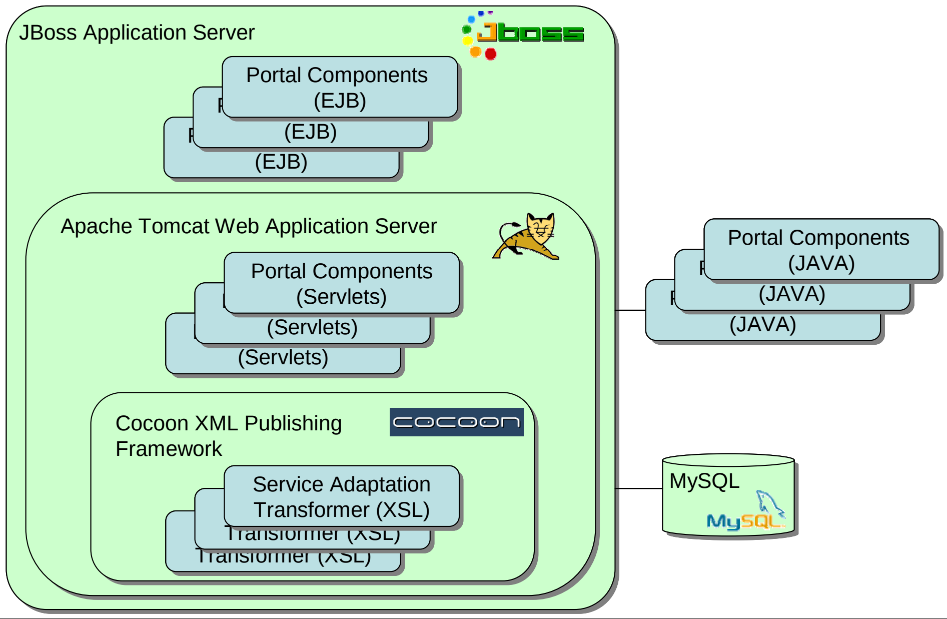


Figure 4-6: Technology Viewpoint

As shown in Figure 4-6, the individual portal components are realized either as external Java programs, as Enterprise Java Beans, as servlets, or as XSL style sheets. Despite of the different platforms, all components communicate with each other mainly using the Java RMI protocol. The references of the components are registered and can be obtained therefore from the Java Naming and Directory service (JNDI) managed by JBoss[36]. Additionally, the JBoss application server maintains a connection to a MySQL[37] database to store the persistent data. This database connection is also used by the other components through the provided JDBC drivers to store their own data.

Generally, the implementations do not use functions, which are unique to the selected implementation and execution platforms, and can therefore be executed on different but compatible systems.

4.4 General Component Model

This section introduces the components of the I-centric User Interaction portal. The components as shown in Figure 4-7 realize altogether the different functions specified by the model to realize I-centric User Interaction.

[36] The JBoss Application Server is one of the most popular Java application server in the industry [WWWJBOSS]

[37] MySQL Open Source Database [WWWMYSQL]

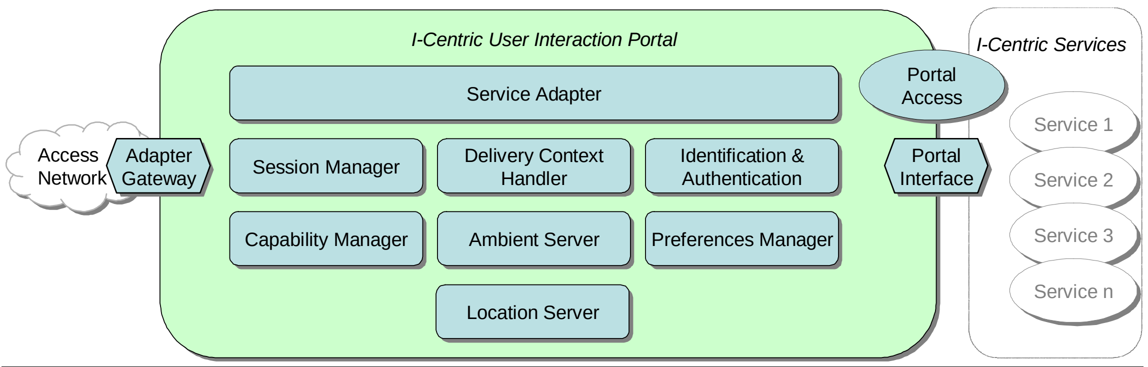


Figure 4-7: Component Model of the I-centric User Interaction Portal

The **Adapter Gateway** is responsible to connect the portal to the respective access networks and communication services. It supports the required protocols and is able to deliver the adapted user interface through the network to the user's terminal and to receive the user's responses, which are forwarded to the Service Adaptation Function. The Adapter Gateway also tries to capture as much as possible information about the Access Mechanism, such as capabilities of network and terminal, as well as ambient information. There are several implementations of Adapter Gateways for different supported communication services, i.e. one for telephony-based access and another one for WWW-based access.

The **Session Manager** manages the individual sessions necessary to realize the interaction between user and services. There is the Access Session describing the connection between terminal and portal and there is the Service Session describing a concrete service usage. The underlying session concept is an elementary prerequisite in order to realize advanced interaction schemas such as multi-modal user interaction, i.e. to interact with the service using multiple different terminals parallel or to suspend a service usage for later resumption eventually on a different terminal.

The **Capability Manager** captures and manages the capabilities of the Access Mechanism. This includes the properties, features, and characteristics of the user's terminal as well as of the respective access network. The Capability Manager provides the necessary information to adapt the presentation of a service to the suitable format enabling best user experience. For example, the components of the Service Adaptation Functions interpret the information provided by the Capability Manager in order to select and, if necessary, to convert the media elements provided by the service.

The **Location Server** is able to provide position data of users and of terminal devices. It is much related to the Ambient Information Server, but concentrates only on location information. For the realization of location-based services, the Location Server supports a number of different positioning technologies, such as GPS, cell-based (in wLAN and GSM), as well as the traditional active badge systems for indoor and short range positioning. The Location Server also realizes a special gateway to other existing position data collecting systems. [HÜNERBERG] describes the developed 'location aware service and application platform', which has been used for the realization of the location server, more detailed.

The **Ambient Server** manages ambient information according to the Ambient Awareness model introduced in section 3.4. It captures data provided by sensor networks, from the location server, as well as from the Adapter Gateways. Furthermore, it implements some revaluation functions, which interpret the raw sensor data and derive higher-level presentations needed for a reasonable adaptation of the service behavior and the service presentation.

The **Preferences Manager** manages the preferences of users. According to the Personalization model introduced in section 3.3, the Preferences Manager stores the preferences data in a structured format, which enables the user to specify the preferences for services dependent on certain conditions, i.e. by defining a Selection Context. The preferences are to be considered in the adaptation of the service user interface and in the execution of the service. In the complete interaction chain between user and service, there are several decisions to be made, e.g. which terminal and which media is preferred for push delivery. These decisions should preferably consider the user's preferences in order to depict a real personalized system.

The **Delivery Context Handler** provides all data to describe the current context, which the interaction takes place. It includes the information about the Access Mechanism as provided by the Capability Manager, the ambient information as provided by the ambient server, and the user's preferences retrieved from the preferences manager. The Delivery Context Handler realizes a facade object to these individual components enabling a comfortable access to the Delivery Context data. This means, the clients, which are interested in this information, such as the services and the Service Adaptation Function, do not have to ask all the respective components directly, but can obtain the complete Delivery Context from the Delivery Context Handler at once. The Delivery Context Handler creates and manages virtual contexts based on the available information.

The **Identification and Authentication** component enables to identify the user, who is requesting the portal services. Identification is the prerequisite for personalization, but also for ensuring access restrictions. There are multitudes of different approaches to obtain and to verify the identity of a user. The different approaches also provide the identification with different confidence and trustfulness. For example, the identification of the user can be derived from the telephone number of the mobile phone, by which the user is accessing the portal. However, it cannot be ensured that it is really the user, who owns the phone actually. Therefore, the Identification and Authentication component implements a number of different technologies to obtain the user's identification. It also considers the available ambient information, e.g. if one user is located nearby a terminal and there is no other user in the surrounding, it can be assumed that it is this user, who is accessing the system, if there comes a request from this terminal.

The **Service Adapter** component implements the Service Adaptation Function as specified by the Generic User Interaction model (section 3.5.5) and depicts the core function to realize the device independent user interaction. It transforms the user interfaces described in the Generic User Interaction Markup Language (GUIML) into the specific representations as required by the user's terminal under consideration of the Delivery Context data. It also transforms the user's feedbacks from the technology specific format into the general presentation as defined by the portal interfaces in order to be passed to the respective service. The Service Adaptation Function processes the content as provided by the service in several steps, denoted as the transformation pipeline (see section 3.2.5.1).

The **Portal Interface** depicts the access to the portal from external components, such as the services. On the one hand, the interface provides the exchange of GUIML documents, i.e. the service user interfaces, and according feedbacks of the user. On the other hand, the interface enables the services to access the Delivery Context data and the Preferences Management System to adapt their behavior according to the ambient information and to the user preferences. Using this interface, the service can also store preferences data for the user, which were specified during the service interaction.

The **Portal Access** represents a special portal service. It belongs to the portal platform and has therefore complete access to all the components. When the user establishes a connection to the portal, this special service is called first. There can be different Portal Access services for the individual users or one Portal Access service, which behaves according to the user's preferences, i.e. as specified by according preferences. If the user could not be identified in the Access Session directly, the Portal Access service takes care about the request and provides reasonable choice, such as to login to the system or to start another service anonymously. If the user is already identified in the Access Session, the according user preferences are looked up and the Portal Access react appropriately, such as to present the user's personal start page with links to the services. The Portal Access service may also resume a suspended session, if specified by the user. This means, this service is the only component of the portal platform visible to the user. It implements all functions necessary for users in a personalized and user-friendly way hiding all technical aspects of the portal platform.

4.5 Adapter Gateways

On the one hand, the Adapter Gateways transform the network and device specific requests into the general representation understandable by the other portal components. On the other hand, the Adapter Gateways capture as much as possible information from the Access Mechanism, such as the capabilities and characteristics of terminal and network, available ambient information, but also user preferences stored on the user's terminal. As shown in Figure 4-8, the Adapter Gateway uses several components of the portal framework to realize the exchange of the content, but also to provide additional data relevant to describe the terminal connection.

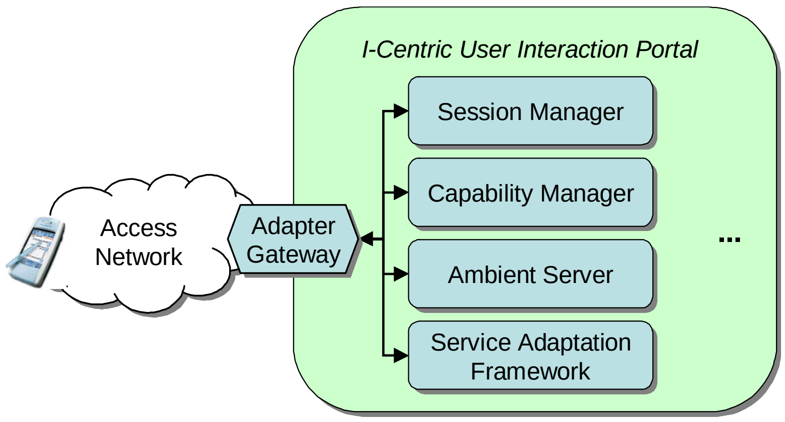


Figure 4-8: Interaction of the Adapter Gateway with Portal Components

Figure 4-9 describes an incoming call[38] scenario. When receiving a new request from the network (1), the Adapter Gateway has to establish an Access Session first (2). The Adapter Gateway has to determine, whether the incoming request is a new request or whether the request belongs to an existing Access Session. Based on the Access Session it provides further data captured from the request to the other components (3+4). If the request does not indicate explicitly the service to connect to, the default address as defined in the Access Session is used by the Adapter Gateway. This address describes a complete request through the Service Adaptation Framework. Accordingly, the Adapter Gateway executes the request and receives content to be presented to the user (4+5). The returned content is converted into the suitable format by the Service Adaptation Framework, i.e. according to the Access Mechanism and terminal capabilities. The Adapter Gateway sends the content to the terminal and waits for the response (6+7). The response is again a request, which is forwarded to the Service Adaptation Function (8).

[38] The term 'call' is used here to denote any kind of incoming request including messages, WWW, and WAP connection requests.

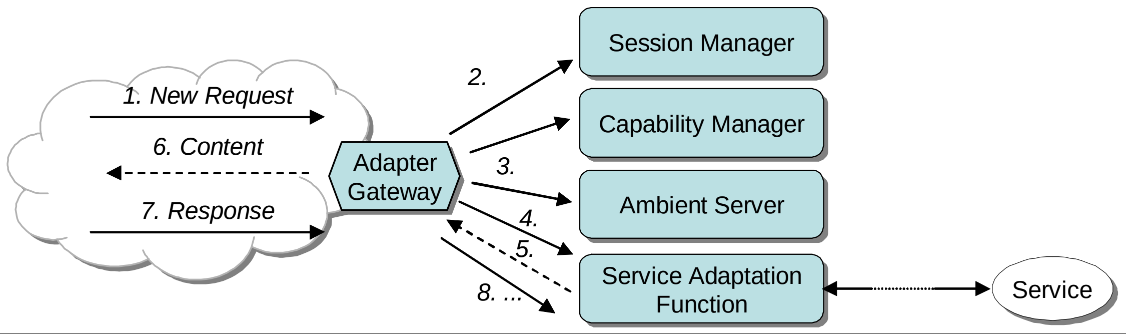


Figure 4-9: Function of the Adapter Gateway

If the user explicitly ends the interaction by closing the terminal connection, the Adapter Gateway requests the Session Manager to conclude the Access Session accordingly. For example, in WWW or WAP there is no possibility to close a connection. The Adapter Gateway itself has therefore to determine, if the terminal connection is not used anymore. Therefore, it applies a timeout mechanism. If there are no responses from the user for the given timeout value, the session is regarded as being closed and the Adapter Gateway requests to conclude the corresponding Access Session.

Beside the incoming call[38] scenario, the Adapter Gateways support outgoing connections, likewise. For example, in a push scenario the service wants to delivery a message to the user or wants to interact with the user. Accordingly, it can request the portal either to send a message or to establish a connection to the user. The service has only to specify the user, the kind of content delivery, i.e. message or interaction, and the actual content. The content is described by a request reference in the same way as in the user-initiated scenario. In this way, the Service Adaptation Function can retrieve and adapt the content, likewise. Only, the initial request differs in the two scenarios. The portal evaluates the user’s preferences and the available ambient information to determine the most suitable terminal according to the demanded kind of delivery. For example, from the user’s location available terminals can be revealed. Additionally, the user’s preferences should indicate the user’s preferred terminal, such as the personal mobile phone to be selected.

According to the selected terminal, the Adapter Gateway initializes a connection to the terminal and starts the interaction by presenting the content retrieved through the Service Adaptation Function. For message-based delivery, the Adapter Gateway first retrieves the content from the Service Adaptation Function by specifying to target format suitable for the messaging system. Then, it creates the message including the returned content and delivers it to the corresponding messaging center, i.e. SMS center or mail server.

For the support of the WWW, WAP, telephony, and massaging-based interaction, a number of respective Adapter Gateways were implemented. They behave as described above. The following sections describe the technology-dependent differences for the three kinds of Adapter Gateways.

4.5.1 WWW and WAP Adapter Gateway

The WWW and WAP Adapter Gateway is the simplest Adapter Gateway. The Service Adaptation Function provides the content described in HTML and WML. Therefore, the WWW and WML Adapter Gateway just has to forward the content to the accessing terminal and to forward the received response request accordingly to the Service Adaptation Function. Special attention has to be put on the management of the Access Session. Usually, WWW as well WAP are stateless technologies. However, by applying suitable session concepts, the individual communication requests can be mapped to an Access Session. The Adapter Gateway, therefore, assigns a session identifier to the content send to the terminal in order to determine the corresponding Access Session. If possible, the cookie-based session management is used, otherwise, according session identifier are added to all the links contained in the document. In this way, the Adapter Gateway recognizes to which session an incoming request belongs.

When an HTTP request from a WWW or WAP browser is received, which cannot be assigned to an existing Access Session, the Adapter Gateway request the Session Manager to create a new Access Session. The corresponding Access Session identifier is used in the following in the communication with the respective terminal, i.e. stored in a cookie on the terminal or used as a parameter in the links. The Adapter Gateway uses a pre-defined timeout indicating when non-active sessions should expire. This means, if there are no responses coming from the terminal for a certain amount of time, the session is regarded as expired and correspondingly the Service Session Manager is requested to conclude the Access Session. Further request, coming from the terminal would initiate a new Access Session, which possibly requires the user to login again.

When a new Access Session has to be created, the Adapter Gateway also checks, whether it can retrieve additional information possibly relevant to the Capability Manager as well as the Ambient Information System. For example, the user agent string contained in the header of the HTTP request reveals some indications of characteristics of the terminal. In this way, it can be determined for example, whether the terminal has a full size display or just a small display like PDAs or mobile phones. This is possible, because the browser applications are diverse on the different terminal platforms. The user agent string has therefore to be provided to the Capability Manager as a minimum in order to derive suitable adapted content presentations.

Furthermore, by interpreting further HTTP header data and by using small Java scripts, additional relevant data can be obtained. If the browser supports and allows the execution of the Java scripts, the screen resolution and other parameters can be captured in this way. Therefore, the Adapter Gateway infiltrates the actual content with such scripts, which cannot be recognized by the user.

At the time of writing this thesis, there were no real CC/PP-based terminals available. However, CC/PP should be the preferred solution to derive the capabilities of the terminal. The actual implementation supports already the CC/PP-based negotiation of capabilities, which was also test with specially developed browser applications. However, this depicts more or less only a feasibility study and is not relevant for practical systems.

The WWW and WAP Adapter Gateway is realized as a servlet in the WWW application server. All requests to this server are routed through the Adapter Gateway servlet by an according specification of servlet context mapping.

4.5.2 Telephony Gateway

The telephony gateway enables the connection of telephony system to the I-centric User Interaction portal. This means, users can call the portal with their ordinary telephone in order to start a speech-based interaction with the services. Similarly, the portal can also call the user to establish the speech-based interaction or at least to deliver a voice message.

The telephony gateway supports speech-based user interaction. The interaction is described in VoiceXML, which is a language to describe voice dialogs. The services adaptation function creates according VoiceXML documents, which are to be processed by the telephony gateway. Therefore, the telephony gateway consists of a VoiceXML interpreter, speech converter, and a Mediagate.

The VoiceXML interpreter processes the VoiceXML document as received from the Service Adaptation Function and provides the according prompts and input possibilities to the presented to the user. The speech converters are able to execute speech synthesis and speech recognition functions, i.e. to translate pure text into speech and vice versa. The Mediagate is a gateway to telecommunication networks. It supports to receive and setup call connections, as well as to record and to include the audio data into the connections, i.e. to record the user’s speech and play back the portal’s audio content including the synthesized speech.

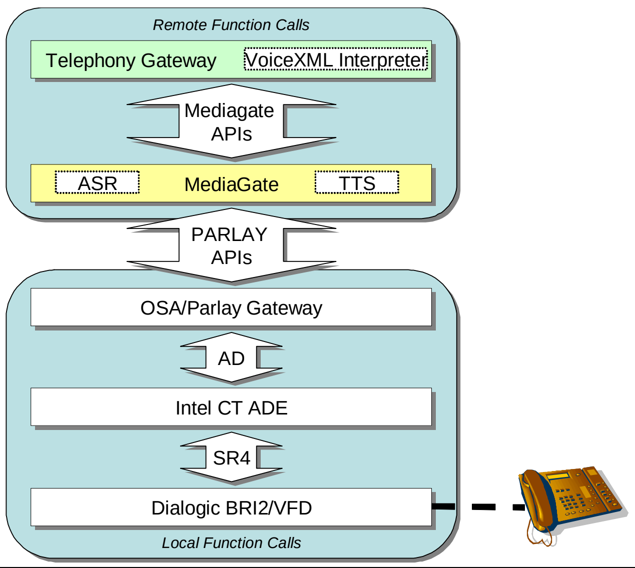


Figure 4-10: The Structure of the MediaGate

The Mediagate used in the realization of the I-centric User Interaction portal includes text-to-speech and automatic-speech-recognition functions. It signals incoming calls to the telephony gateway, which request the Session Manager to create a new Access Session. According to the Access Session, or rather to the Service Session identified in the Access Session, the telephony gateway requests the Service Adaptation Function to provide the content to be presented to the user. This content is described in VoiceXML. Accordingly, the telephony gateway requests the VoiceXML interpreter to process the document and to determine the concrete dialog, which has then to be played by to the user. This means, the VoiceXML interpreter picks the complete document into small pieces, which each describe one dialog, i.e. output and possible input.

This dialog fragment is then forwarded to the Mediagate, which uses its text-to-speech function in order to obtain the synthesized speech. The resulting audio data are played back to the user by sending the audio data to the stream associated with the terminal connection using the user-interaction interfaces of the Parlay gateways. The Mediagate starts also immediately waiting for input, i.e. to enable barge-in input. The input received from the Parlay gateway can be speech and DTMF tones. The speech is translated into text by using the automatic speech recognition function. According to the dialog specified by the VoiceXML interpreter, the recognized inputs are matched to the possible inputs, i.e. according to the grammar as specified in the dialog fragment. The input is then send back to the telephony gateway, which itself forwards the input to the VoiceXML interpreter. The VoiceXML interpreter can now continue the processing of the VoiceXML document, which can lead to the next dialog fragment or to a new request to the services. Accordingly, the telephony gateway submits the next fragment to the Mediagate or executes the request to the service through the Service Adaptation Function starting the complete loop again.

If the user ends the connection, the Mediagate signals correspondingly the termination to the telephony gateway, which in turn frees its resources and request the Session Manager to conclude the corresponding Access Session.

The telephony gateway is realized as an external Java program, but is to be integrated in the JBoss application server by transforming the gateway’s implementation into an Enterprise Java Bean. It uses CORBA for the communication with the Mediagate and Java RMI for the communication with the relevant portal components.

4.5.3 Messaging Gateways

The messaging gateways supported the retrieval and the sending of instant messages (IM), short massages (SMS), multimedia messages (MMS), and e-mails. Because, messaging is an asynchronous communication service, the messaging gateway does not to maintain an Access Session. It receives a complete request to send a message from the portal. This request includes the address, the message type to be used, i.e. IM, SMS, MMS, or e-mail, and the request to the service content. The messaging gateway, therefore, first executes the service request, which returns the content to be included in the message. The returned content suits to the specified message type, i.e. n*160 characters for SMS and IM, SMIL document for MMS, plain text or HTML document for e-mail.

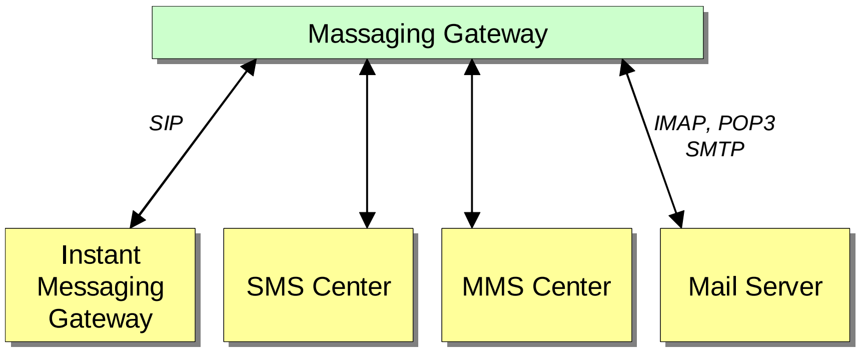


Figure 4-11: The Messaging Gateway

The messaging can now simple contact the according messaging service centers and send the content to the specified address. It returns the result code, i.e. message was taken, delivered, or read, to the portal component, which has initiated the sending of the message.

The retrieval of messages was still under development at the time of writing this thesis, but it is going to be similar. After message retrieval, an according request is created containing the addresses and the message content. This request is forwarded to an according portal component, which then analyses this request and determines whether it fits to an existing Service Session or whether a new Service Session has to be created. Accordingly, the request is transformed by the Service Adaptation Function to the representation as required by the respective service, i.e. GUIML, and delivered to this service.

The messaging gateway and corresponding components, such as the connectors to the different kinds of message centers, are realized as Enterprise Java Beans running in the JBoss application server.

4.6 Service Adaptation Component

The Service Adaptation component implements the Service Adaptation Function as specified by the model for Generic User Interaction (section 3.5.5). It transforms the content provided by the services into the specific representation required by the Access Mechanisms. This means, the GUIML documents are transformed into the respective user interface languages, such as HTML, WML, VoiceXML, and message formats. The complete adaptation is executed in several transformation steps, denoted as the transformation pipeline[39]. The individual steps consider the capabilities of the Access Mechanism and the user preferences in the transformation. The documents passed from one transformer step to the next one are XML-based. Therefore, the Service Adaptation component is realized using an XML publishing framework. Such XML publishing framework provides the functionality needed to realize an XML transformation pipeline by executing a chain of XSL transformations. Several XSL transformers are executed inside the XML publishing framework as depicted in Figure 4-12.

[39] Introduced in section 3.2.5.1

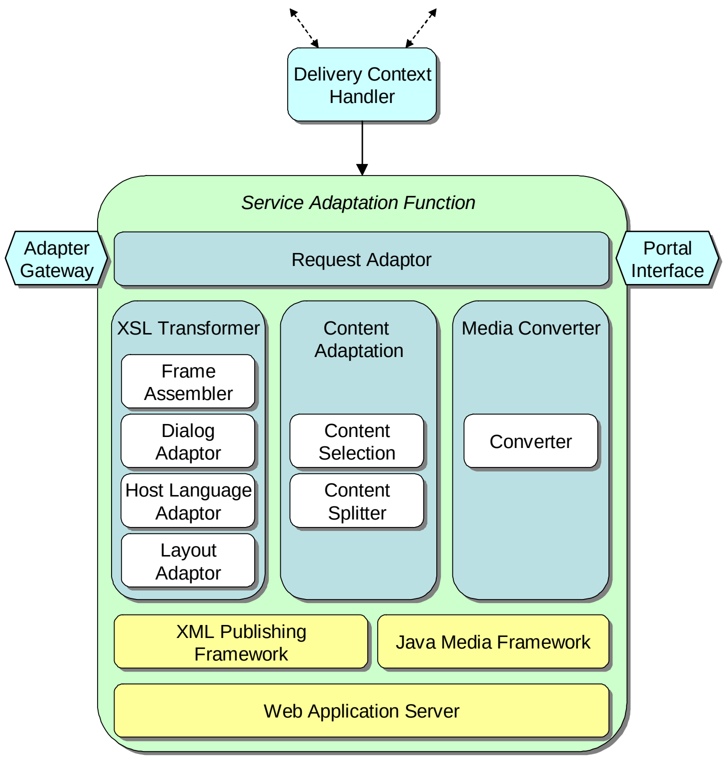


Figure 4-12: Service Adaptation Framework Architecture

The Apache Cocoon XML Publishing Framework [WWWCOCOON] was selected as the base of the whole Service Adaptation Function. This framework offers the functionality needed to execute multiple transformations in a pipeline. The pipeline can consist of internal Cocoon transformers and self-developed transformers. One important internal transformer is the XSL transformer needed for most of the transformation steps of the Service Adaptation Function. Based on the SAX [WWWSAX] transformer, it is a straightforward task to provide own transformers as needed for the Service Adaptation, which can be integrated into the Cocoon transformation pipeline.

Apache Cocoon is running inside the Apache Tomcat Server [WWWTOMCAT] as a servlet application. Tomcat is the servlet container, which is used in the official reference implementation for the Java Servlet and JavaServer Pages technologies [SUNJSPSPEC]. Since Apache Cocoon imposes Java [SUNJAVASPEC] as technology, the Java Media Framework [SUNJMFSPEC, SUNJMFGUIDE] was selected for the analysis and conversion of media objects in the realization of the media converter. The media converter is implemented as a servlet running in the Tomcat web server.

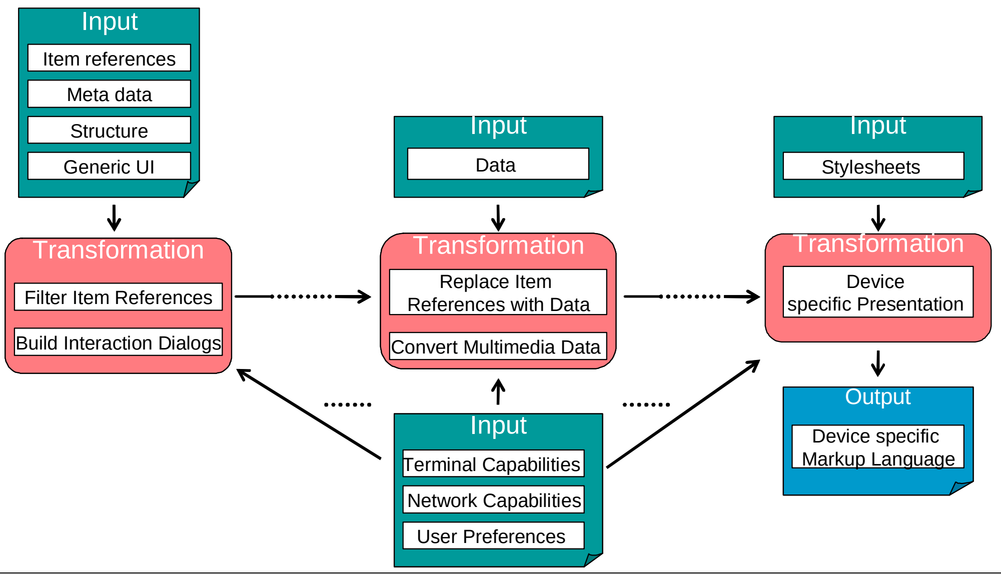


Figure 4-13: Transformation Pipeline Overview

As shown Figure 4-13, the content received from the service is transformed in several transformation steps, which are realized as a pipeline of respective transformations. The following sections will introduce the individual steps and their realization in the Cocoon framework more detailed.

4.6.1 Transformation Pipeline in the Apache Cocoon Framework

The transformation pipeline is defined by the Cocoon sitemap. The part of the sitemap, which selects the pipeline of transformations to execute depending on the user agent identifier, is a sitemap resource with the name 'devindp_external_service'. It uses the Cocoon browser selector for the selection of further sitemap resources for different user agents. The 'devindp_external_service' resource is called through a general matcher.

```
<map:match pattern="*/**">
 <map:call resource="devindp_external_service">
  <map:parameter name="target" value="{2}" />
  <map:parameter name="serviceid" value="{1}" />
 </map:call>
</map:match>
```

The user agent identifier is provided by the WWW browser application or by the respective Adapter Gateway. The resources called depending on the user agent identifier are:

- service_wml
- service_vxml
- service_html

The resources include all transformation steps to obtain the device dependent format. The URL of the actual service is retrieved by the getRealServiceURL action and written to a sitemap parameter. Thus, it can be passed to all transformers needing it. The action gets the URL by accessing the service deployment descriptor. In this way, the service identifier can be mapped to the corresponding URL. When the request of the client arrives, the request URL is parsed to obtain:

- The service identifier
- The actual service context, which is requested
- Internal ‘GET’ parameters for the Service Adaptation Function
- ‘GET’ parameters for the service

After the necessary data are obtained, the transformation pipeline is started by calling the resources as specified in the sitemap.

4.6.2 Request Adapter

First, the actual URL of the service and the service parameters are passed to the request adapter, which is realized as a Cocoon generator. It depicts the start of the pipeline. The corresponding class implementing the component is included in Cocoon with the name ‘requestadapter’ as shown in following example:

```
<map:generator  name=”requestadapter”
     src=”adaptation.generation.RequestAdapter”/>

<map:generate type=”requestadapter” src=”{target}”>
   <map:parameter name=”serviceid” value=”{serviceid}” />
   <map:parameter name=”service_deployment_descriptor”
        value=”file://services.xml” />
</map:generate>
```

It retrieves the GUIML content from the service. The type of the request, which can be ‘GET’ or ‘POST’ has to be maintained when passing the request to the real service URL. Therefore, the request adapter distinguishes between these two types. For ‘GET’ requests, all request parameters are parsed and added to the URL. For ‘POST’ requests, all parameters are included in the body.

Because the format of the request parameters passed by the accessing devices is not homogenous for multiple values, the request adapter converts parameters of VoiceXML and WML clients to the format common for WWW clients. VoiceXML clients pass multiple values for one attribute name distinguished by a point, and WML clients use a semicolon. They are converted to multiple attributes with the same name.

4.6.3 Frame Assembler

The frames referenced in the document provided by the services are included into the document via the frame assembler. It first transforms the referenced frames into ‘xinclude’ instructions by an according XSL transformer. This transformer searches all frame tags with an ‘href’ attribute replaces them with:

```
<xi:include href=”{@href}” xml:base=”{$realserviceurl}”/>
```

The variable ‘$realserviceurl’ contains the URL of the service. If the service author adds an ‘xml:base’ attribute to the frame tag, this URL is taken instead of the URL of the hosting server:

```
<xi:include href=”{@href}” xml:base=”{@xml:base}”/>
```

Afterwards, the Cocoon ‘XInclude Transformer’ inserts the referenced contents, i.e. by the ‘xi:include’ statement, into the document. The sitemap fragment looks as follows:

```
<map:transform src=”stylesheets/includeframecontent.xsl”>
 <map:parameter name=”realserviceurl” value=”{realserviceurl}” />
</map:transform>

<map:transform type=”xinclude” />
```

4.6.4 Content Adaptation

The content adapter is realized as a SAX transformer component, which is defined as the ‘mediaadapter’ in the Cocoon sitemap:

```
<map:transformer name=”mediaadapter”
 src=” adaptation.transformation.MediaAdapter” />

<map:transform type=”mediaadapter”>
 <map:parameter name=”realserviceurl” value=”{realserviceurl}” />
</map:transform>
```

The real service URL is needed to resolve the relative media element references. The content adapter selects the appropriate media elements depending on the capabilities of the accessing device and the user’s profile and preferences. Therefore, the content adapter component includes the content selection function, which selects content depending on the user’s profile and preferences. Because it is a SAX transformer component, the methods of the media adapter are called when SAX events occur. When the ‘startElement()’ method is called, it is checked if the element is a SMIL element. The content adapter handles SMIL elements only. All other elements are ignored, i.e. passed unchanged from the content adapter transformer to the next component in the pipeline.

When a SMIL test attribute is found, the component evaluates if the attribute matches with the capabilities of the device and the user’s profile and preferences. Therefore, the Delivery Context data are requested from the Delivery Context Handler. If the value of the test attribute equals the value as specified in the Delivery Context, it evaluates to true and the content is not removed of the document. When it is evaluated to false, flags are set to remove all SAX events from the document until the closing tag of the element is found.

The SMIL switch statement, which contains a list of alternative media elements, is traversed until an appropriate media element is found. When a media element without a surrounding switch statement occurs, it is treated like an element without any alternative. For every media element found in the document, an instance of the class ‘MediaElement’ is created. The content adapter tries to determine the type of the media element and analyses as much information as possible about it. The method ‘matchesCapabilities()’ of the ‘MediaElement’ checks if the device is able to display the media element. Certain user preferences are also taken into account. When the method returns true, i.e. the media element matches the capabilities, the rest of possible alternative media elements in the document is ignored and it is inserted into the document.

Thereby the content adapter inserts an inline representation of the media element reference if necessary. When the media element does not match the capabilities, the method ‘achievableMatchCapabilities()’ is called to test if the media element could be converted to suit the capabilities of the Access Mechanism. When this method returns true, ‘convertConflictingProperties()’ of ‘MediaElement’ is called which creates a list of conversion parameters for the call of the media converter. The call to the media converter servlet with its parameters is written into the document instead of the static media reference. Otherwise, the text representations of the media elements are inserted, if these are provides by the service.

The media converter is implemented as a servlet in the WWW application server in form of an on-the-fly converter. It depicts is a Cocoon reader component, which reads a media element from a given URL, and returns a converted media element in the body of the request:

```
<map:reader name=”ontheflyconverter”
      src=”adaption.reading.OnTheFlyConverter” />
<map:match pattern=”ontheflyconverter.*”>
   <map:act type=”request”>
      <map:parameter name=”parameters” value=”true” />
      <map:read type=”ontheflyconverter” src=”{url}” />
   </map:act>
</map:match>
```

The URL, the target media type, and several conversion parameters are passed as request parameter to the servlet. The type of the media element is determined by the file extension. The type of the media returned by the converter can be different. The WWW application server has to allow changes of the content type. For every media type, there is a corresponding code segment trying the conversion to meet the requested properties. The following conversion possibilities have been implemented:

- JPEG, GIF, and WBMP media type conversions in any direction
- Changing the size of the picture while maintaining its aspect ratio
- WAV to AU media type conversion
- Conversion of encoding, sample rate, and the number of channels for audio files

The addition of further conversion possibilities is very simple by extending the implemented media converter. For example, the Java Media Framework can handle many types of media and provides many functions for their manipulation. It was selected for the current implementation of the media converter component.

The media type of the source media element is determined through its suffix. Depending on the media type, different algorithms for image and audio conversion are selected. If a new conversion should be added, a new algorithm has to be incorporated. First, the media format conversion is performed to achieve the desired properties is done first. Afterwards, the media element is converted to a new media type if this is requested through the ‘mediaType’ parameter. To convert JPEG or GIF images to WBMP images, the color palette has to be adapted to monochrome. Therefore, the color histogram is calculated and a threshold value is used to determine the pixels that should be converted to white and those that should be converted to black.

4.6.5 Dialog Adapter

For every target device there is a different XSL style sheet doing the dialog adaptation. The style sheets have to adapt the XForms model and the single XForms controls to the dialog format of the target device. The corresponding sitemap entry is a call to the Cocoon XSL transformer as follows:

```
<map:transform src=”stylesheets/xforms2{device}.xsl” />
```

Suitable transformers have been implemented for HTML, WML, and VXML. Accordingly, the device variable in the sitemap can be ‘html’, ‘wml’, or ‘vxml’. The main problem of adapting the XForms model is to select predefined values of attributes, so that they are only passed as parameters when they are not used in any XForms control. Therefore, the whole document has to be scanned for XForms controls belonging to the matched model. For HTML devices, the predefined values are passed as hidden parameters when no XForms control exists that uses the attribute name. WML and VoiceXML clients use the definition of variables. Multiple values defined for an XForms instance variable have to be converted to client specific representations, too.

Another problem is the selection of the XForms controls belonging to their model. They have to be selected because all belong to one form, whereas other controls may belong to another form. All XML elements being child of one model to the beginning of the next model have to be selected. It is not possible to take the first XForms control as the start of the form and allow defining the XForms model anywhere in the document. The reason for this is that the first XForms control could be on another level than the last one, making it impossible to surround it by a form tag, as needed for HTML. For example, the following complex selection expression selects all controls belonging to a model:

```
<xsl:apply-templates select=
     "following-sibling::*[generate-id(following-sibling::xfm:model[1])
     = generate-id(current()/following-sibling::xfm:model[1])]"/>
```

However, there is still a grouping problem. An ‘xfm:model’ element is found and all the elements before the next ‘xfm:model’ element should be selected. Thus, for all the nodes in the group that should be selected, the next xfm:model element is going to be the same element. The next ‘xfm:model’ element is found with the XPath expression:

```
following-sibling::xfm:model[1]
```

The next elements of any kind are found by

```
following-sibling::*
```

Now a predicate is needed, which identifies all those elements, whose next ‘xfm:model’ element is the same as the next ‘xfm:model’ element for the current ‘xfm:model’ element. Within the for-each construct the ‘xfm:model’ element is the current node, so the next ‘xfm:model’ element can always be identified with:

```
current()/following-sibling::xfm:model[1])
```

Nodes can be compared with ‘generate-id()’ function of XPath. All XForms controls have their own template for the adaptation, as shown in the following example:

```
<xsl:template name="xformcontrols">
   <xsl:choose>
      <xsl:when test="name()='xfm:input'">
         <xsl:call-template name="input"/>
      </xsl:when>
      <xsl:when test="name()='xfm:secret'">
         <xsl:call-template name="secret"/>
      </xsl:when>
      <xsl:when test="name()='xfm:textarea'">
         <xsl:call-template name="textarea"/>
      </xsl:when>
      <xsl:when test="name()='xfm:submit'">
         <xsl:call-template name="submit"/>
      </xsl:when>
      <xsl:when test="name()='xfm:select1'">
         <xsl:call-template name="select1"/>
      </xsl:when>
      <xsl:when test="name()='xfm:select'">
         <xsl:call-template name="select"/>
      </xsl:when>
      <xsl:when test="name()='xfm:trigger'">
         <xsl:call-template name="trigger"/>
      </xsl:when>
   </xsl:choose>
</xsl:template>
```

Section 7.2 contains an overview of the templates for the user interface technologies HTML, WML, and VoiceXML. These templates are used in the implementation of the Dialog Adapter component to transform the XForms controls in the respective representation of the target user interface technology.

For HTML and WML, the adaptation is simple. To get an idea of how such a transformation looks like, an example for the XForms input control transformation is given in the following. For HTML, the XForms input control is adapted to an HTML input control with the attribute type=’text’. The predefined value must be inserted as value attribute. It is selected from the ‘ref’ attribute of the XForms control. The label of the XForms control can be either an XML element or characters. This is distinguished through the ‘xsl:choose’ expression at the beginning of the template.

```
<xsl:template name=”input”>
   <xsl:variable name=”ref” select=”@ref” />
   <xsl:variable name=”model” select=”@model” />
   <xsl:choose>
      <xsl:when test=”xfm:label/*”>
         <xsl:copy-of select=”xfm:label/*” />
      </xsl:when>
      <xsl:otherwise>
         <xsl:value-of select=”xfm:label” />
      </xsl:otherwise>
   </xsl:choose>
   <input type=”text” name=”{@ref}”>
      <xsl:if
         test=”//xfm:model[@id=$model]/xfm:instance/*[name()=$ref]”>
         <xsl:attribute name=”value”>
           <xsl:value-of
           select=”//xfm:model[@id=$model]/xfm:instance/*[name()=$ref]” />
         </xsl:attribute>
      </xsl:if>
   </input>
</xsl:template>
```

For WML, the transformation is simpler, because the attribute has already been set to the predefined value when the model was transformed. The according XSL transformation for WML looks as follows:

```
<xsl:template name=”input”>
   <p>
      <xsl:choose>
         <xsl:when test=”xfm:label/*”>
            <xsl:copy-of select=”xfm:label/*” />
         </xsl:when>
         <xsl:otherwise>
            <xsl:value-of select=”xfm:label” />
         </xsl:otherwise>
      </xsl:choose>
      <input name=”{@ref}” />
   </p>
</xsl:template>
```

In this document, only the examples above are presented. However, the transformation of other XForms controls for HTML and WML look similar.

The VoiceXML transformation needs more complex algorithm. Predefined values impose the inclusion of an extra question to the user that asks him if he wants to change the input. This is done through a template called ‘change_predefined’. Extra text, which is inserted into the document, e.g. to prompt the user for input, has to consider the language of the user. The language is derived from the user’s preferences. However, the adaptation of input controls is done analogue to the HTML and WML approach. In the following example, a VoiceXML field is used to gather input from the user, the prompt tells the user the label of the XForms control and asks for ‘input?’ in the selected language. The user is requested to enter some digits, because the field is of type digits.

```
<xsl:template name="input">
   <xsl:variable name="model" select="@model" />
   <xsl:variable name="ref" select="@ref" />
   <xsl:call-template name="change_predefined">
      <xsl:with-param name="model" select="@model" />
      <xsl:with-param name="ref" select="@ref" />
   </xsl:call-template>
   <field name="{@ref}">
      <xsl:attribute name="type">digits</xsl:attribute>
      <prompt>
         <xsl:value-of select="xfm:label" />
         <xsl:text>. </xsl:text>
         <xsl:value-of
            select="$phrases/phrase[@key=&#39;input&#39; and
                   lang($language)]"/>
      </prompt>
      <help>
         <xsl:value-of select="xfm:hint" />
      </help>
      <xsl:call-template name="catchnoinput" />
      <filled>
         <xsl:call-template name="saythankyou" />
      </filled>
   </field>
</xsl:template>
```

All links, menus, and navigation among frames (change of the scope) have to be made globally available by defining events, because the user may always request the available links, menus, and frames. Thus, events with grammars for every single link are inserted. The grammars have to be converted to lower case to be recognized as whole word by the VoiceXML interpreter. If it is necessary for the VoiceXML interpreter, a DTMF grammar is inserted, too. The DTMF grammar specifies the possible options. Accordingly, the possible input consists of two digits, the first digit for the general type of input and the second digit for the concrete input:

- 6 (M) for menu options
- 7 (S) for changing the scope
- 5 (L) for following a link

For example, the user has to press keys ‘5’ and ‘2’ to follow link one. The numbers are read to the user with the names of the options when he requests help. The following example shows a template, which inserts the according grammars for all links:

```
<!—targets of the actual frame -->
<xsl:for-each select="frame[@id=$frame]">
   <xsl:call-template name="insertLinksToAllTargets" />
</xsl:for-each>
<xsl:template name="insertLinksToAllTargets">
   <!—simple links -->
   <xsl:for-each select=".//xfm:trigger[xfm:action/xfm:load]">
      <link next="{xfm:action/xfm:load/@xlink:href}">
         <grammar type="application/x-gsl">[(
            <xsl:value-of
               select="translate(xfm:label,$ucletters,$lcletters)" />
         )]</grammar>
      </link>
   </xsl:for-each>
   <!—select1 links -->
   <xsl:for-each
      select=".//xfm:select1/xfm:item[xfm:action/xfm:load]">
      <link next="{xfm:action/xfm:load/@xlink:href}">
         <grammar type="application/x-gsl">[(
            <xsl:value-of
               select="translate(xfm:label,$ucletters,$lcletters)" />
         )]</grammar>
      </link>
   </xsl:for-each>
</xsl:template>
```

For every frame, the adaptation has to insert all links. The template ‘insertLinksToAllTargets’ is called. This template inserts the VoiceXML element link for every XForms trigger and select1 control that loads a URL. The grammar is set to the label of the trigger or select1 after converting is to lower case and removing special characters. The XForms controls for these navigation controls are then removed of the dialog flow.

4.6.6 Host Language Adapter

The host language adapter is an XSL transformation with different style sheets for each target device:

```
<map:transform src=”stylesheets/page2{device}.xsl” />
```

For HTML clients, the transformation is very simple. The only task is the replacement of SMIL references. The WML and VoiceXML transformations on the other hand have to remove nesting XHTML paragraphs of the document. The WML language knows some of the XHTML tags, so that a one-to-one conversion is possible. The remaining XHTML elements have to be mapped to those, which have a functional similarity. For VoiceXML clients, all XHTML elements are mapped to prompts to the user.

4.6.7 Layout Adapter

Like the dialog adapter and the host language adapter, the layout adaptation also needs single XSL style sheets for each device:

```
<map:transform src=”stylesheets/frames2{device}.xsl”>
```

HTML clients are able to display a complex layout when the display size is large enough. Thus, a layout configuration file is used to render the frames to a page layout done with HTML tables. Each table cell is filled with the content of the frame with the corresponding id.

WML clients can only display one frame at a time. References to the other frames are inserted into the options menu.

VoiceXML clients can also only present one frame at a time to the user. The links to the other frames were already inserted in the dialog adaptation. Here, only a ‘goto’ statement to the selection of the next link, menu, or frame is added to the end of the page.

The cascading style sheets, which are considered as a part of the layout adapter, are applied by the client itself. The corresponding cascading style sheet file is read from the layout configuration file and inserted into the document.

4.6.8 Finalization of the Pipeline

Finally, the transformation pipeline ends with a serializer transforming the XML document into device specific formats. Cocoon internal components are used for this:

```
<map:serialize type=”wap” />
```

There are serializers for HTML, WML, and VoiceXML. The serialized result depicts now a document completely conformant to the respective specification of the target languages, i.e. HTML, WML, and VoiceXML. All elements of the Generic User Interaction language are removed. Standard-compliant clients can now process the resulting document in order to present the corresponding user interface to the user.

4.7 Session Manager

The Session Manager maintains the access and the Service Sessions centrally. The Session Manager is always requested, when a new session is to be created or an old session is to be concluded. It uses a unique identification scheme to identify each session. When a session is being set up, a corresponding session object is created and registered in the session database. There are different kinds of session objects for the Access Sessions and for the Service Sessions. This session object contains all the relevant data necessary to describe the session, i.e. session identifier, service identifier, Delivery Context identifier, etc. The Access Sessions describe the connection of a terminal to the I-centric User Interaction portal and the Service Session describes a service execution. Both sessions are always linked to an individual user, i.e. to the user, who accesses the portal and who uses the service. The user can have multiple connections to the portal concurrently, which results in multiple different Access Sessions of the user. The user can also use multiple services concurrently, which results in multiple different Service Sessions. Both types of sessions are separated, but can be linked to each other by references.

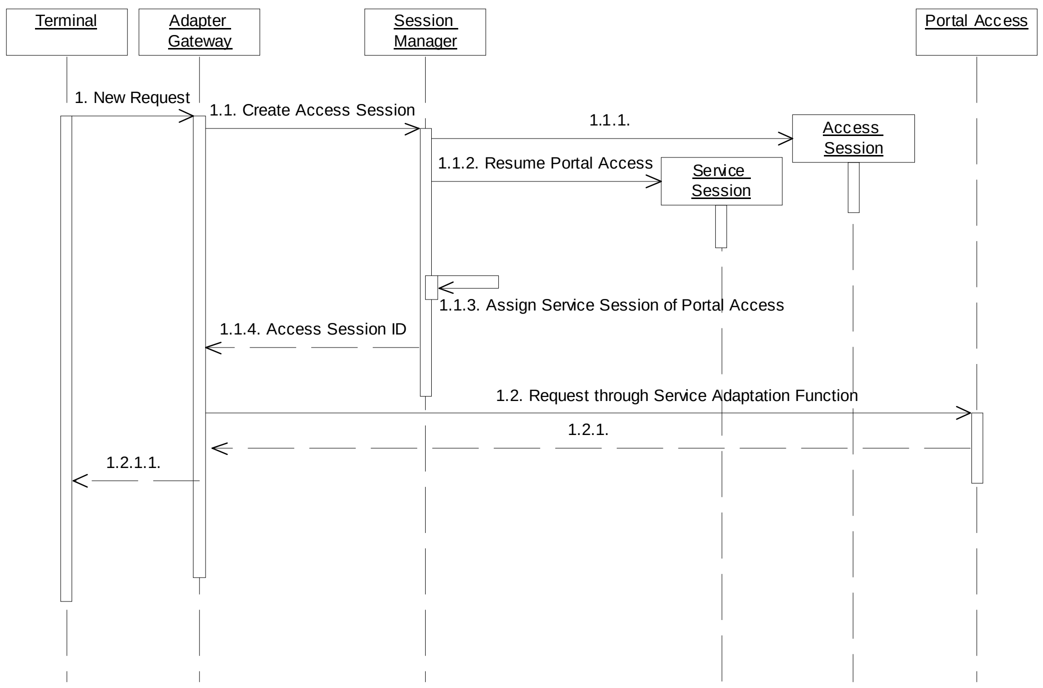

Terminal
Adapter Gateway
Session Manager
Portal Access
1. New Request
1.1. Create Access Session
1.1.1.
Access Session
1.1.2. Resume Portal Access
Service Session
1.1.3. Assign Service Session of Portal Access
1.1.4. Access Session ID
1.2. Request through Service Adaptation Function
1.2.1.
1.2.1.1.

Figure 4-14: Creation of initial Access Session

When the user accesses to the portal, i.e. starts the interaction, the newly created Access Session contains a reference to the special Portal Access service as shown in Figure 4-14. This service, described in section 4.14, depicts the user's initial 'homepage' and provides some general administrative functions as well as the start of a service usage according the user's preferences. When the user selects a service and starts the interaction, an according Service Session is created or resumed as well as assigned to this Access Session. When the user ends the interaction, the Access Session is closed, but the Service Session can remain, i.e. suspended.

It depends on the service, whether it makes use of the suspension feature or not. Preferably based on the user's preferences, the service configures the Service Session to be non-persistent, i.e. to be concluded when there no Access Session assigned anymore, or to be persistent, i.e. to suspend the Service Session when the assigned terminal session is closed. When a new Access Session was created and the user selects a service to interact with, the Portal Access service searches the session database whether there is already a corresponding Service Session. If there is one, this session is resumed and assigned to the new Access Session. The session of the Portal Access service is suspended again.

Whereas the access is specific to the terminal connection, the Service Session and the Delivery Context are related to a user. Therefore, it is necessary to identify the user in order to select the suitable Service Session and Delivery Context, which might already exit. The Adapter Gateway therefore passes as much as possible information about the Access Mechanism to the session manager. This information can include the address of the terminal, such as the telephone number, or a user identifier as derived from the WWW browser by the Adapter Gateway.

The Session Manager requests the Identification and Authentication component, which tries to map the available data to a user identifier. If no real user can be determined, the default identifier ‘anonymous’ is returned. Accordingly, the Service Session of the Portal Access service and the corresponding Delivery Context are newly created or reused, if they already exist for the respective user. The sessions include then the user identifier and the user’s preferences can be retrieved accordingly from the Preferences Management System.

4.8 Identification and Authentication Component

The Identification and Authentication component provides on the one hand the identification and on the other hand the authentication of users. The identification can be derived from a number of data provided by the Access Mechanism, such as the address of the localization of the user’s terminal. Depending on the concrete application scenario, it can be sufficient to personalize the behavior based on such data. For example, if the access comes from a terminal, which belongs as a personal device to a specific user, the system may assume that this user is accessing the portal.

The identification is sufficient to provide personalized appearances. However, this assumption can be wrong and the identification therefore depicts only a guess and is not trustful. Because of this, the additional authentication is required, which obtains the user’s identification in a trustful manner. Based on the user’s authentication, the portal and the services can rely on the identification.

As described in section 4.7, the Session Manager needs to obtain the user identifier in order to look up the corresponding Service Sessions belonging to the user and to look up the corresponding Delivery Context. Accordingly, it provides the parameters as captured from the Adapter Gateway. The identification function interrogates the Preferences Management System and the Delivery Context Handler to find a mapping to a corresponding user. For example, from the number of the telephone is searched in the user database to find out, whether this is a personal device belonging to a user. If a corresponding user is found, the corresponding user identifier is returned assuming that this user has initiated the call.

The identification function characterizes the results found with probability values, because based on the available data, the user’s identification can only be guessed. Basically, the certainty depends on the quality of data provided by the Adapter Gateways. For example, the telephone could be used by another person, which would lead to a wrong identification. Because of this, the identification function provides rather indications of the user’s identification, which are therefore classified as being uncertain. Services should not trust on identification with a low certainty if they are going to provide sensible data. However, considering the certainty carefully this kind of identification is sufficient for the basic personalization. In order to achieve the authentication of the user, the Portal Access service provides an according login mechanism, where the user can authenticate by providing the user identifier and a corresponding password. This results in a reliable identification.

The Identification and Authentication component provides several approaches for the identification. Beside the database look up, it can also evaluated certificates, which might be provided by the terminal. In this way, the portal platform provides multiple possibilities to identify the accessing user and to ensure the derived identity. The current implementation supports the mapping of terminal and network addresses as well as the temporary storage of user identification on terminal using cookies mechanism. It is going to be extended to support also speaker recognition, certificate, and smart card based identification, likewise.

The Identification and Authentication component is realized as an Enterprise Java Bean running in the JBoss application server.

4.9 Capability Manager

The Capability Manager component manages the capabilities of the Access Mechanism. The Adapter Gateways are responsible to gather and to provide as much as possible information about the capabilities. These are sent to the Capability Manager, which analyses, revaluates, and stores them associated to the Access Session identifier. For example, the WWW Adapter Gateway can easily obtain the user agent identifier from the HTTP header, but possibly no further information about the actual device. The user agent identifier is specific to the individual WWW browser applications and can additionally reveal the operating system. It can be used therefore to conclude the actual capabilities of the terminal, even if these are not provided directly.

For example, the Capability Manager can easily differentiate between regular computers and PDAs, which discloses different screen resolutions relevant for the Service Adaptation. Similarly, according to the network address of the accessing terminal, the capabilities of the access network can be obtained. For example, from terminal’s IP address it can be determined whether the terminal is in a GPRS or local network.

In this way, the Capability Manager can determine additional characteristics of the terminal and therefore enrich the available information of the Access Mechanism. Basically, it maintains a database of capabilities sets related to corresponding conditions, i.e. to the user-agent identifier or to the network. This database has to be maintained and partially configured manually by the portal platform provider, e.g. definition of different local networks with different capabilities. The raw data provided by the Adapter Gateways are compared by the Capability Manager with the database and the capability set describing a specific terminal connection is extended in this way.

Fundamentally, this approach depicts a temporary solution as long as no mechanisms are provided by the terminals and the networks directly, such as CC/PP. However, the current solution already enables reasonable results and represents a suitable alternative for the time being. The current implementation of the Capability Manager is open and flexible to integrate different approaches easily. For the management of the user agent identifier in the HTTP protocol it uses the DELI implementation, which is an open source Delivery Context Java library supporting CC/PP and UAProf (with HTTP header negotiation) [WWWDELI]. The component is implemented as an Enterprise Java Bean running in the JBoss application server.

4.10 Ambient Server

The Ambient Server implements the functions of the Ambient Awareness model as specified in 3.4.4. It retrieves data from a sensor network consisting of several smart IP devices based on the TINI architecture [WWWTINI]. These sensor devices are able to record temperature and brightness values as well as to detect active badges in their surrounding. The Ambient Server manages the received ambient data and executes reasonable interpretation and revaluation algorithms. For example, the identification number of the active badge and the according sensor device, which has detected the badge, is transformed to a user and location relationship. In this way, the Ambient Server provides user’s current position in a technology independent format.

The ambient data provided by the Ambient Server are to be included in the Delivery Context and therefore requested by the Delivery Context Handler. According to the location of interest as specified by the Delivery Context Handler, the Ambient Server collects relevant ambient data and returns an according ambient profile in an XML document using the specification of the ambient information data model (section 3.4.3). The Ambient Server is implemented as an external Java program and runs independently from the portal.

4.11 Preferences Manager

The Preferences Manager implements the necessary function to manage personal preferences of users. The preferences are structured according to the specification of the Personalization model and described in section 3.3.3. Furthermore, the Preferences Manager provides the internal and the external interface as introduced in section 3.3.6. This component is realized as an Enterprise Java Bean and runs in the JBoss application server. The preferences data are persistently stored in the corresponding MySQL database. The implementation follows the functional decomposition as proposed by the Personalization model. There are no access control mechanisms implemented for the internal access assuming that only internal components belonging to the trustworthy domain can access the preferences manager. The access to the underlying database, i.e. the MySQL database, is also restricted by the provided security concepts of MySQL. The external access, i.e. for services, is realized in the portal by the portal interface, which performs the general access control.

4.12 Delivery Context Handler

The Delivery Context Handler maintains the Delivery Contexts containing all data necessary for the service behavior and Service Adaptation. A Delivery Context is always associated with one user and possibly with an Access Session of this user. It contains therefore the user identifier, the user’s location, the ambient information available for this location, the preferences of the user, and the capabilities of the Access Mechanism. The Delivery Context does not necessarily be assigned to a user’s Access Session. This is necessary to support services, which wants to get into contact with the user even when the user does not have a connection to the portal, i.e. service-initiated interaction. If there is no Access Session assigned, the Delivery Context does not contain the data provided by the Capability Manager. Therefore, the Delivery Context Handler provides a method to retrieve the Delivery Context data belonging to an Access Session and a method to retrieve the Delivery Context data belonging to a user. The second method is to be used, if there is no Access Session available, and the resulting Delivery Context does not contain any capabilities of the Access Mechanism accordingly. If there is an Access Session used in the current service execution, i.e. there is an interaction between user and service, the first method should be used, which returns a complete Delivery Context including the capabilities of the corresponding Access Mechanism.

The Delivery Context Handler implements only the internal interface, which is specified in section 3.2.6, because the corresponding external interface is realized by the Portal Interface (section 4.13). For the arrangement of a Delivery Context, this component interrogates the Capability Manager, the ambient server, and the Preferences Manager to collect the relevant data. The Delivery Context Handler component is realized as an Enterprise Java Bean running in the JBoss application server.

4.13 Portal Interface

The Portal Interface provides the external access to the portal. This means the services use this interfaces in order to access specific portal functions. The actual content, i.e. the GUIML documents, is exchanged via the HTTP protocol. This means the Service Adaptation Function, which requests the content from the services in order to adapt and to delivery it, uses directly the

HTTP protocol to interact with the service. From the service's point of view, the Service Adaptation Function looks like a normal WWW browser application apart from the requested content type, which is GUIML instead of HTML.

Furthermore, the services need to access the Delivery Context data and to manage the user's service preferences. Therefore, a suitable interface is provided by the Portal Access component. Fundamentally, any kind of remote procedure call (RPC) protocol can be applied for the realization of this interface. Because, the services are already web services supporting the HTTP protocol, the SOAP protocol was chosen for the actual implementation. However, the design of this Portal Interface component allows attaching any further protocol adapter, such as for example CORBA or pure Java RMI. The interface in the current implementation uses the Axis framework, which is an open source implementation of the JAX-RPC API from Apache[40]. It realizes the basic concepts of handlers and type-mappings driven by a lightweight SAX-based processing engine. Axis comes with a dedicated deployment format called Web Service Deployment Descriptor (WSDD), which configures the runtime engine with new request/response flows, service providers, and type-mappings.

The API of the Portal Interface is specified in WSDL (see appendix section 7.5.1) and is provided to the service developers. These can easily create corresponding stubs to be used in the implementation of the services. Usually, the various available implementations of web services engines provide suitable compilers to obtain Java stubs from the interface specification. Then, the service developers can easily use the generated Java classes in the service implementation, e.g. in Java Server Pages or servlets. Of course, there are also other compilers supporting different platforms, such as .Net, if the service developers prefer these.

Figure 4-15 shows the methods defined in the portal interfaces.

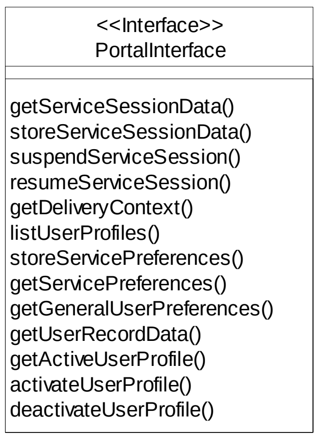


Figure 4-15: The Portal Interface

The corresponding Java interface looks as follows:

```
package ICUI;
import java.io.*;

class Attribute {
   String name;
   Serializable value;
}

public interface PortalInterface {
```

[40] See [WWWAPACHE]

```
   public Attribute[] getServiceSessionData(String
      serviceSessionIdentifier);
   public Boolean storeServiceSessionData(String
      serviceSessionIdentifier,
      Attribute[] sessionData);
   public Boolean suspendServiceSession(String serviceSessionIdentifier);
   public String resumeServiceSession(String serviceSessionIdentifier);
   public String getDeliveryContext(String serviceSessionIdentifier);
   public String[] listUserProfiles(String serviceSessionIdentifier);
   public Boolean storeServicePreferences(
      String serviceSessionIdentifier,
      Attribute[] attributeList, String userProfileName);
   public Attribute[] getServicePreferences(
      String serviceSessionIdentifier);
   public Attribute[] getGeneralUserPreferences(
      String serviceSessionIdentifier);
   public Attribute[] getUserRecordData(String serviceSessionIdentifier);
   public String getActiveUserProfile(String serviceSessionIdentifier);
   public Boolean activateUserProfile(String serviceSessionIdentifier,
      String userProfileName);
   public Boolean deactivateUserProfile(String serviceSessionIdentifier);
}
```

Each method has a ‘serviceSessionIdentifier’ parameter. This is necessary, in order to enable the Portal Interface component to obtain the corresponding information, such as user identifier to interrogate the Preferences Manager as well as to request the Delivery Context. There are special attributes in the Service Session object, which cannot be overwritten by the service, e.g. the user identifier. These attributes are just ignored in the ‘storeServiceSessionData’ method.

Furthermore, the Portal Interface also performs an access control and security checks by implementing an authentication function. Requests coming from certain components, especially external components like services, are identified according to the applied communication protocol and their access to functions and data of the portal platform can be granted or forbidden. Therefore, the accessing services have to be authenticated first. This is realized by two alternatives. In the first and insecure option, it is assumed that only the real service knows the Service Session identifier specified in the individual method calls and each method call is therefore trusted. However, if the identifier is intercepted by a malicious service, the correct access control cannot be ensured. That is why the services should really authenticate ensuring their real identification. Therefore, the Portal Interface uses secure HTTP for transmission and X.509-based certificates for authentication. In this way, the Portal Interface can claim the identification of the calling component by asking for the certificate, which is also used for the encryption. The certificate and therefore the identification can be scrutinized by the authentication function. These are common security mechanism in HTTP-based communication system and supported by almost all web services frameworks.

The Portal Interface component is realized as an Enterprise Java Bean running in the JBoss application server. It uses the Axis web service container for the communication with the external services.

4.14 Portal Access Service

The Portal Access service is a special service operated by the portal provider. It belongs to the group of internal components and has therefore the corresponding access rights to the portal components. This service manages the access of the user to the portal. It depicts a kind of ‘homepage’ for the users, which can be personalized. This service provides the necessary functions to the users to manage their preferences and their service usage. The user can specify which services should appear in which order as links or as inline frames on the home page. Whereas the external services provide the actual benefit to the user, this services depicts the comfortable portal to these external services.

If the user could not be identified though data captured from the Access Session, the user can authenticate to the system by corresponding login function provided by the Portal Access service. When the user selects a specific service, the Portal Access service determines available suspended Service Sessions, which are reactivated then accordingly in order to start the service interaction. The portal service passes the control to the user while remaining available through suitable links in the interaction, i.e. the user can always return or switch back to the portal service. Therefore, the portal service runs permanently for the user as long there is an Access Session. If there is no Access Session, i.e. the user does not have a connection to the portal, the Portal Access service is suspended until a new Access Session is created.

As stated above, the Portal Access service should appear according to the user's preferences. However, the system provides to operate different kinds of access services. Then, after the user identification is determined, the according access service is selected, which is specified in the general user preferences. The current implementation uses only one Portal Access service, but which can be personalized to suit different requirements of users.

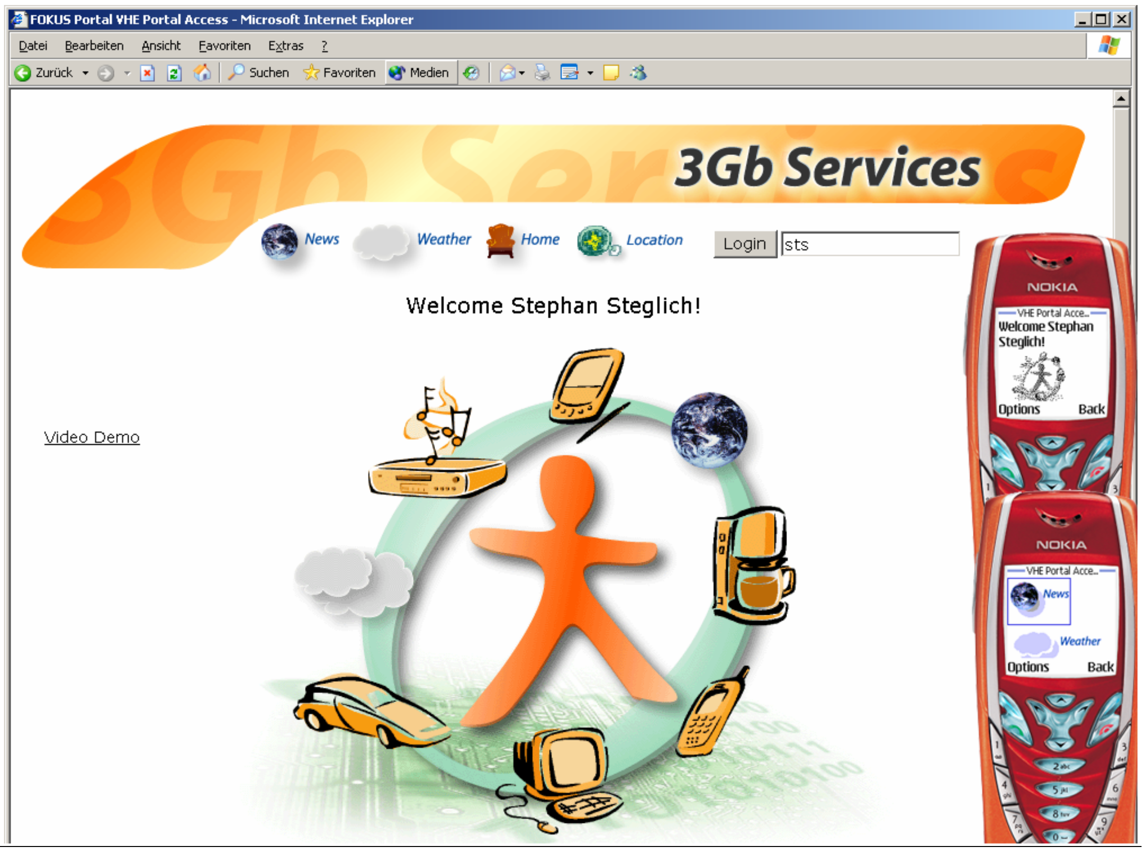


Figure 4-16: Screenshot of the Portal Access Service

4.15 Realized Services

Based on the existing implementation, a number of exemplary services were realized to demonstrate the capabilities and the potential of the I-centric User Interaction approach. The portal supports the speech-based interaction over a telephone connection, the WWW access on normal computers as well as small screen devices such as PDAs, the WAP access from mobile phones, and message-based interaction including short messages, multimedia messages, SIP-based instant messaging, and e-mail. Furthermore, there is a support for service initiated message delivery and user-initiated service interaction. These are the interaction schema, which are used by the realized demonstration services.

Furthermore, the portal supports the gathering and the provisioning of ambient information. This ambient information as well as the user preferences is considered in the service execution as well as in the interaction with the user. Concerning the user preferences, the services itself provide the specification of the user settings during the service usage and store these settings to the Preferences Management System provided by the portal platform. The special Portal Access service provides the necessary support to manage the User Profiles, define the Selection Contexts for each, and to specify the general preferences.

The selected service scenarios are only examples showing relevant aspects and were defined and realized in the course of several research projects. There was a special focus on content-based services as well as on ambient controlling. Accordingly, among the services, which have been realized, the following examples should be introduced in this thesis:

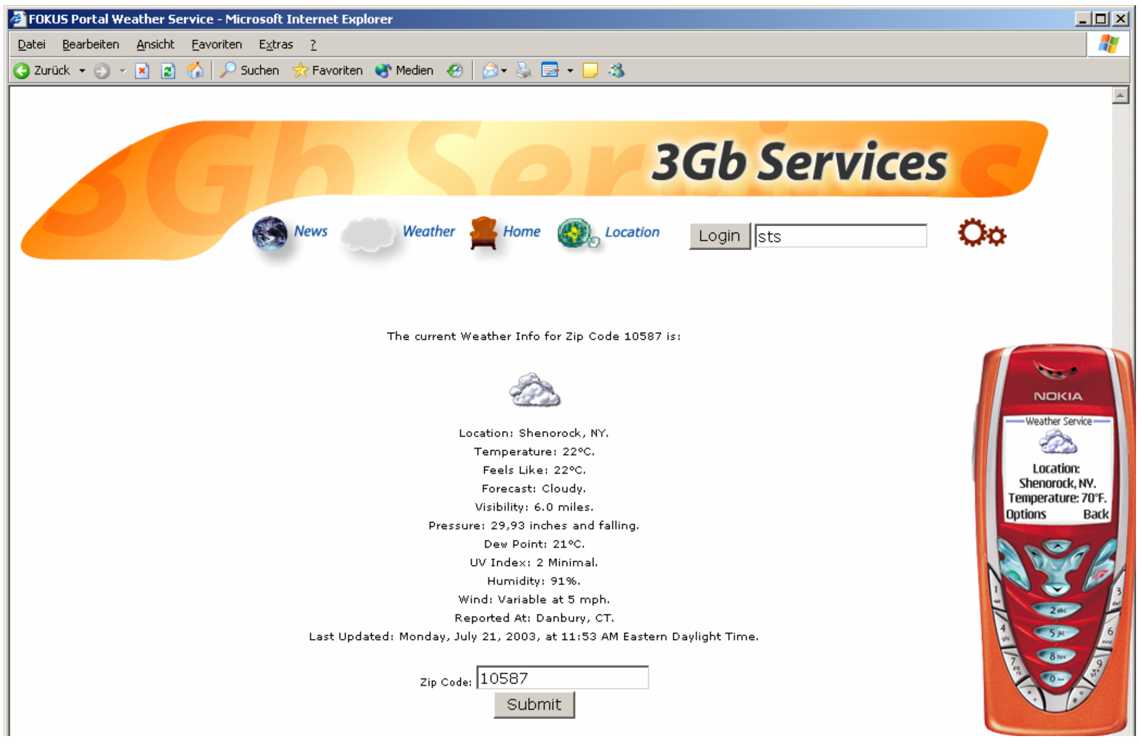
- The Weather service (section 4.15.1)
- The DPA News service (section 4.15.2)
- The Pi-AVIda service (section 4.15.3)
- The Ambient Control service (section 4.15.4)

The following sections describe the individual services and the functions of the portal they use. The introduction concentrates and the main aspects and does not contain technical details.

4.15.1 Weather Service

The weather service is a relatively simple service and was one of the first realized on the portal platform. It uses a selection of public weather forecast services, which are provided as WWW services in the Internet. These services depict the source of weather data. For the purpose of demonstration, the quality of the data is not so important. That is why these selected external services were sufficient for the realized weather service. All the external services provide weather data described in HTML or in XML based on a given location, such as postal code or town name.

Therefore, the weather service asks the portal for available location data of the user in order to request the suitable weather data from the external sources. It also uses the personalization system to define a default or home location, which should be used, if no location data are available. Additionally, the user can specify that the data for the home location are always to be presented supplementary. The service supports the usage from WWW browsers, WAP terminals, and in telephony calls, i.e. speech-based. The goal of this service is to provide the user, a personalized weather forecast accessible at any time and from any device. Figure 4-17 shows a screenshot of the WWW interface of the weather service.


FOKUS Portal Weather Service - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Medien
3Gb Services
News
Weather
Home
Location
Login
sts
NOKIA
Options
Back

The current Weather Info for Zip Code 10587 is:

Location: Shenorock, NY.
Temperature: 22°C.
Feels Like: 22°C.
Forecast: Cloudy.
Visibility: 6.0 miles.
Pressure: 29,93 inches and falling.
Dew Point: 21°C.
UV Index: 2 Minimal.
Humidity: 91%.
Wind: Variable at 5 mph.
Reported At: Danbury, CT.
Last Updated: Monday, July 21, 2003, at 11:53 AM Eastern Daylight Time.

Zip Code: 10587
Submit

Weather Service
Location:
Shenorock, NY.
Temperature: 70°F.

Figure 4-17: Screenshot of the Weather Service

Later, the weather service was extended to delivery messages with the weather forecast to the user. Therefore, the service makes use of the messaging capabilities of the portal. The user can specify in the preferences, whether the service should send the weather forecast at given times or (alternatively or additionally) depending on location change as well as of weather change. In this way, the user could be notified about upcoming weather changes, such as when it is going to rain.

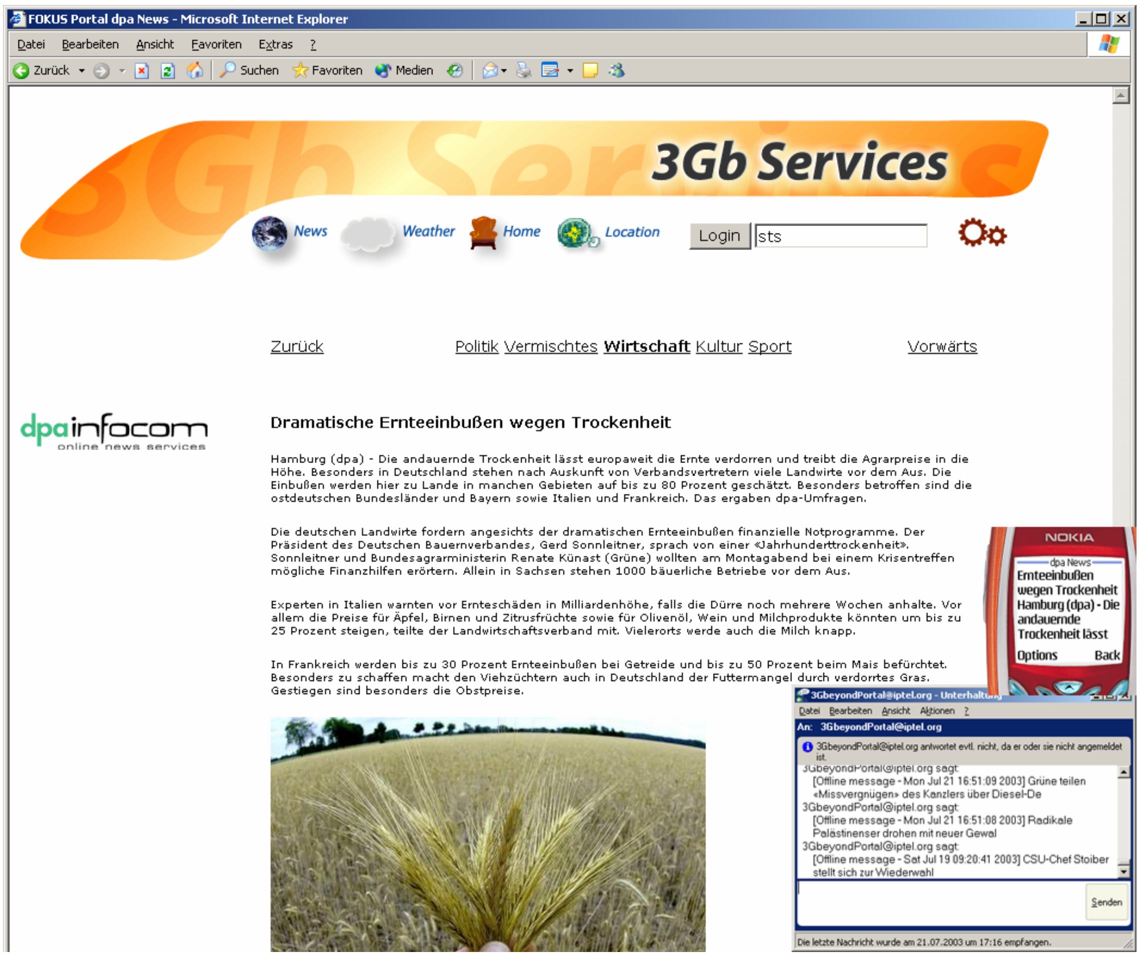
This service considers the location data of the user. Because the current implementation only provides a limited support for the gathering of real location data, there is an additional option to specify the current location manually, by which the automatic recognition can be simulated. The portal supports mainly the indoor localization of users, which is not suitable to this kind of service. The outdoor localization is partially supported, but can be used only in simulations. This is mainly due to the practical restrictions to access the location data available in the mobile telecommunication networks. There is an alternative demonstration using a GPS receiver attached to a PDA, which sends the location data as gathered from the GPS system to the portal's Ambient Server through the corresponding Adapter Gateway.

4.15.2 DPA News Service

The DPA News service is a multi channel online news ticker service. It offers personalized news from different categories, which can be accessed from different kinds of terminals. The service supports the message-based notification of incoming news. The news channels are provided by the dpa-infocom GmbH, which is one of the biggest content providers in Europe delivering news content to numerous media companies. The several content streams are actively sent by an ftp push mechanism. The messages are structured in XML documents containing references to media elements such as pictures, images, and small videos according to the NITF 3.0 standard [NITF]. Accordingly, each message belongs to a certain category and provides some keywords, which are used for the personalized delivery.

The realized service concentrates on the preparation of the news message to be presented through the I-centric User Interaction portal and the delivery through the messaging subsystem of the portal. This means, the service reads the messages when they arrive and creates according

GUIML presentations of each. In this way, the user can access the news messages through a WWW browser, WAP browser, the telephony gateway. Figure 4-18 shows an according screen-shot of the application.


FOKUS Portal dpa News - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Medien
3Gb Services
News Weather Home Location
Login sts
dpainfocom online news services
NOKIA
dpa News
Options Back
3GbeyondPortal@iptel.org - Unterhaltung
Datei Bearbeiten Ansicht Aktionen ?
Senden

Zurück Politik Vermischtes **Wirtschaft** Kultur Sport Vorwärts

Dramatische Ernteeinbußen wegen Trockenheit

Hamburg (dpa) - Die andauernde Trockenheit lässt europaweit die Ernte verdorren und treibt die Agrarpreise in die Höhe. Besonders in Deutschland stehen nach Auskunft von Verbandsvertretern viele Landwirte vor dem Aus. Die Einbußen werden hier zu Lande in manchen Gebieten auf bis zu 80 Prozent geschätzt. Besonders betroffen sind die ostdeutschen Bundesländer und Bayern sowie Italien und Frankreich. Das ergaben dpa-Umfragen.

Die deutschen Landwirte fordern angesichts der dramatischen Ernteeinbußen finanzielle Notprogramme. Der Präsident des Deutschen Bauernverbandes, Gerd Sonnleitner, sprach von einer «Jahrhunderttrockenheit». Sonnleitner und Bundesagrarministerin Renate Künast (Grüne) wollten am Montagabend bei einem Krisentreffen mögliche Finanzhilfen erörtern. Allein in Sachsen stehen 1000 bäuerliche Betriebe vor dem Aus.

Experten in Italien warnten vor Ernteschäden in Milliardenhöhe, falls die Dürre noch mehrere Wochen anhalte. Vor allem die Preise für Äpfel, Birnen und Zitrusfrüchte sowie für Olivenöl, Wein und Milchprodukte könnten um bis zu 25 Prozent steigen, teilte der Landwirtschaftsverband mit. Vielerorts werde auch die Milch knapp.

In Frankreich werden bis zu 30 Prozent Ernteeinbußen bei Getreide und bis zu 50 Prozent beim Mais befürchtet. Besonders zu schaffen macht den Viehzüchtern auch in Deutschland der Futtermangel durch verdorrtes Gras. Gestiegen sind besonders die Obstpreise.

Ernteeinbußen wegen Trockenheit Hamburg (dpa) - Die andauernde Trockenheit lässt

An: 3GbeyondPortal@iptel.org

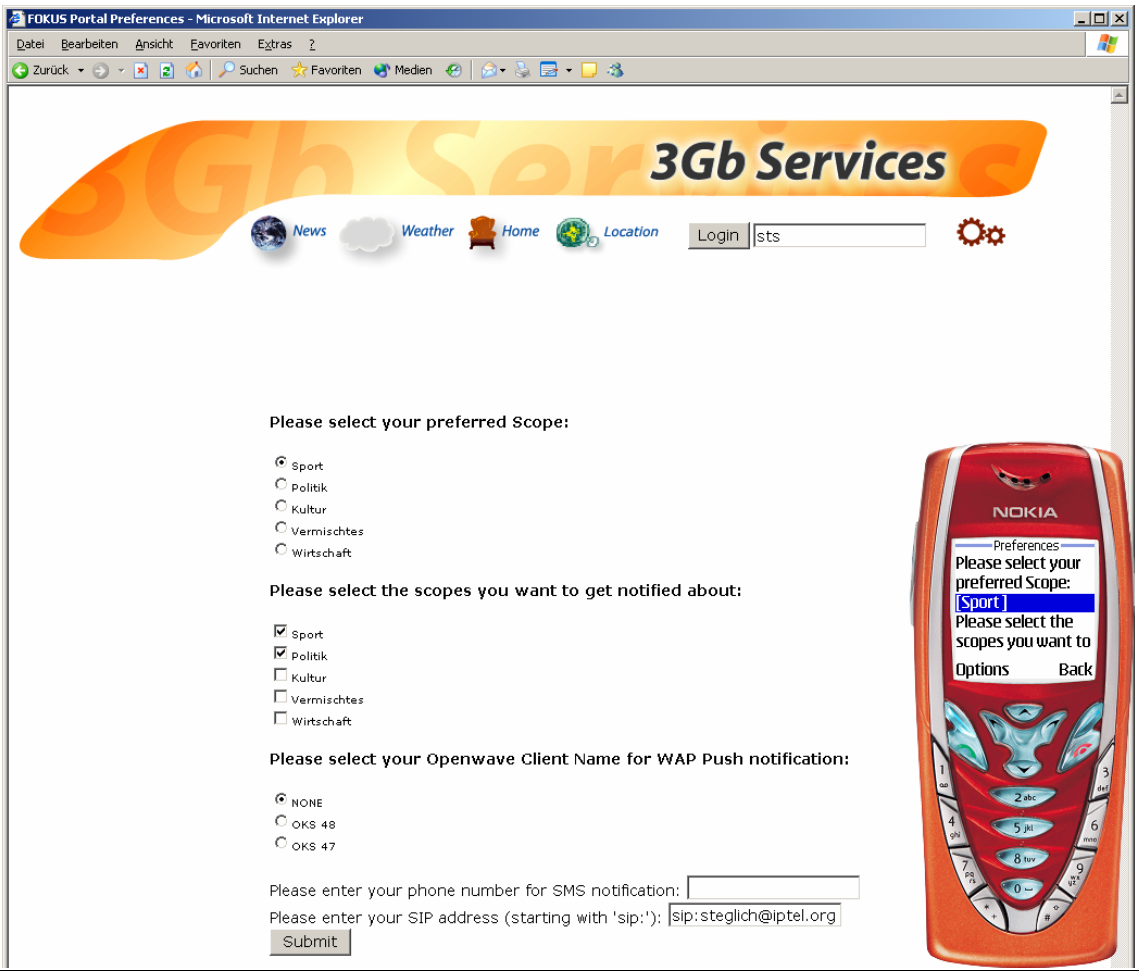
3GbeyondPortal@iptel.org antwortet evtl. nicht, da er oder sie nicht angemeldet ist.

3GbeyondPortal@iptel.org sagt:
[Offline message - Mon Jul 21 16:51:09 2003] Grüne teilen «Missvergnügen» des Kanzlers über Diesel-De
3GbeyondPortal@iptel.org sagt:
[Offline message - Mon Jul 21 16:51:08 2003] Radikale Palästinenser drohen mit neuer Gewal
3GbeyondPortal@iptel.org sagt:
[Offline message - Sat Jul 19 09:20:41 2003] CSU-Chef Stoiber stellt sich zur Wiederwahl

Die letzte Nachricht wurde am 21.07.2003 um 17:16 empfangen.

Figure 4-18: Screenshot of the DPA News Service

Furthermore, the users can select categories and specify the delivery mechanisms in order to instruct the service to send new message accordingly as depicted in Figure 4-19.


FOKUS Portal Preferences - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Medien
3Gb Services
News Weather Home Location
Login
NOKIA
Options Back

sts

Please select your preferred Scope:

[✓ Sport]
Politik
Kultur
Vermischtes
Wirtschaft

Please select the scopes you want to get notified about:

- [x] Sport
- [x] Politik
- [ ] Kultur
- [ ] Vermischtes
- [ ] Wirtschaft

Please select your Openwave Client Name for WAP Push notification:

[✓ NONE]
OKS 48
OKS 47

Please enter your phone number for SMS notification:

Please enter your SIP address (starting with 'sip:'): sip:steglich@iptel.org

Submit

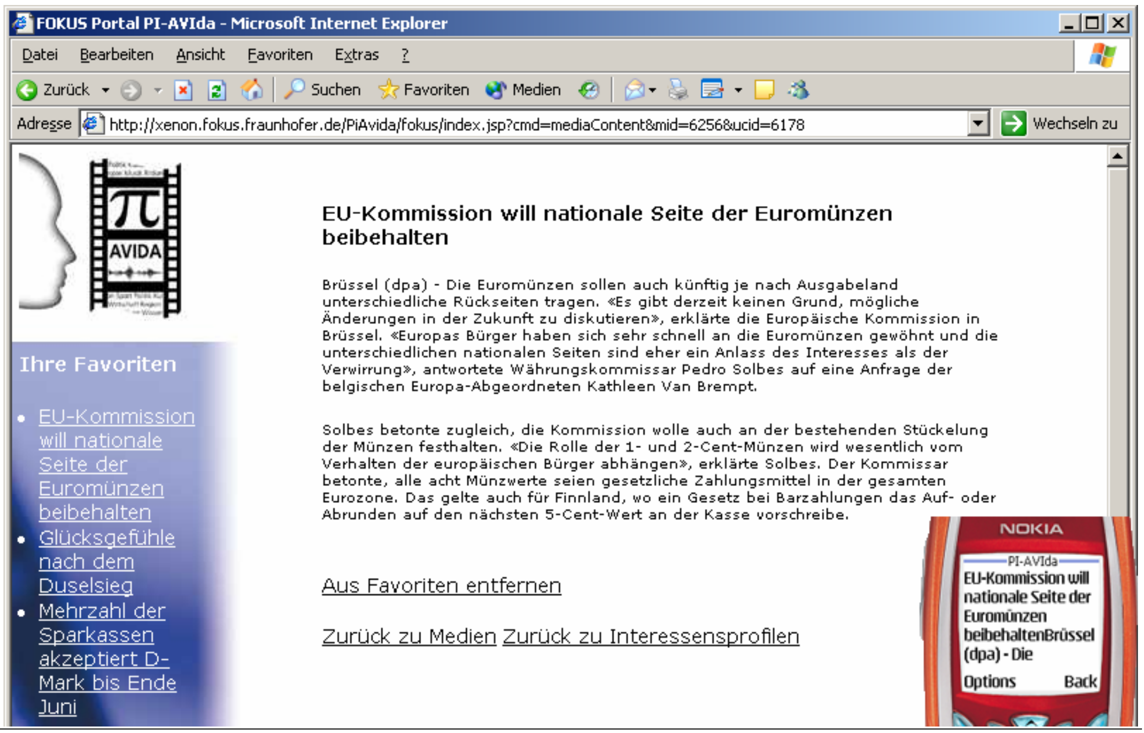
Preferences
Please select your preferred Scope:
[Sport]
Please select the scopes you want to

Figure 4-19: Selection of User Preferences for the DPA News Service

These settings are stored in the corresponding service preferences set. Whenever a new message is retrieved by the DPA news service, it checks all currently valid user preferences and instructs the I-centric User Interaction portal to convert and to deliver the message accordingly. The personalized delivery concentrates first on the selection of the categories used by the content provider, but is being extended by advanced automatic categorization mechanisms. These analyze the content by searching for relevant keywords and use rating methods. In this way, the content to be delivered is selected on the base of given keywords and on base of available rating mechanism. These procedures are subject of a dedicated research project and are already realized exemplary in the Pi-AVIda service (section 4.15.3).

4.15.3 Pi-AVIda Service

The Pi-AVIda service was implemented in the Pi-AVIda project founded by the German ministry of education and research [WWWBMBF]. The goal of the project was to research and develop advanced algorithms of multimedia content categorization and provisioning. The complete Pi-AVIda system maintains a database in which the actual multimedia content as well as the derived categorization information is stored. The Pi-AVIda service provides this content through the I-centric User Interaction portal and uses the available categorization data for the selection and filtering according to the user preferences. Figure 4-20 shows a screenshot of this example application.


FOKUS Portal PI-AVIda - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Medien
Adresse http://xenon.fokus.fraunhofer.de/PiAvida/fokus/index.jsp?cmd=mediaContent&mid=6256&ucid=6178
Wechseln zu
AVIDA
NOKIA
PI-AVIda
EU-Kommission will nationale Seite der Euromünzen beibehaltenBrüssel (dpa) - Die
Options
Back

Ihre Favoriten

- EU-Kommission will nationale Seite der Euromünzen beibehalten
- Glücksgefühle nach dem Duselsieg
- Mehrzahl der Sparkassen akzeptiert D-Mark bis Ende Juni

EU-Kommission will nationale Seite der Euromünzen beibehalten

Brüssel (dpa) - Die Euromünzen sollen auch künftig je nach Ausgabeland unterschiedliche Rückseiten tragen. «Es gibt derzeit keinen Grund, mögliche Änderungen in der Zukunft zu diskutieren», erklärte die Europäische Kommission in Brüssel. «Europas Bürger haben sich sehr schnell an die Euromünzen gewöhnt und die unterschiedlichen nationalen Seiten sind eher ein Anlass des Interesses als der Verwirrung», antwortete Währungskommissar Pedro Solbes auf eine Anfrage der belgischen Europa-Abgeordneten Kathleen Van Brempt.

Solbes betonte zugleich, die Kommission wolle auch an der bestehenden Stückelung der Münzen festhalten. «Die Rolle der 1- und 2-Cent-Münzen wird wesentlich vom Verhalten der europäischen Bürger abhängen», erklärte Solbes. Der Kommissar betonte, alle acht Münzwerte seien gesetzliche Zahlungsmittel in der gesamten Eurozone. Das gelte auch für Finnland, wo ein Gesetz bei Barzahlungen das Auf- oder Abrunden auf den nächsten 5-Cent-Wert an der Kasse vorschreibe.

Aus Favoriten entfernen

Zurück zu Medien Zurück zu Interessensprofilen

Figure 4-20: Screenshot of the Pi-AVIda Service

Another aspect of this project was to obtain suitable categorization data through suitable rating mechanism. The rating of content is realized by explicit and implicit feedback mechanisms. In the explicit feedback, the user can rate the content by specifying a quality value (1-10). In the implicit feedback, the portal analyses, which content was really requested and viewed by the users. This is done by examining the requests to the content coming from the users after they have received the overview of all content elements with some small summaries, i.e. if the users click on the "more..." link. All rating results are provided back to the database, which leads to a refined categorization of the content elements, which is in again used in the following presentations. For example, the users can specify to view only content elements, which have a minimum positive rating.

The Pi-AVIda service uses therefore also the personalization feature of the I-centric User Interaction portal beside the device independent presentation. The personal preferences specify the filtering, the presentation style, and the active delivery of the multimedia content.

The results of the projects with regard to the application of the portal platform are described in [PIAVIDAD5.1, PIAVIDAD5.2, PIAVIDAD5.3, PIAVIDAD5.4].

4.15.4 SDO Service

The SDO service maintains the access to a smart environment system, which consists of Super Distributed Objects (SDO). These objects depict high-level representation of hardware and software elements providing standardized interfaces to be controlled by intelligent services. Besides the controlling by automatic services, the direct control of these objects by the users is relevant, likewise. The corresponding user interface to the SDO system is realized by the SDO service using the I-centric User Interaction portal.

The service enables users to view the status of their environment with any kind of terminal. In this way, the users can monitor relevant environments in different locations and situations, e.g. at the office desk using their laptop but also in the car using their mobile phone with a speech-based access. Figure 4-21 shows a screenshot of this application.

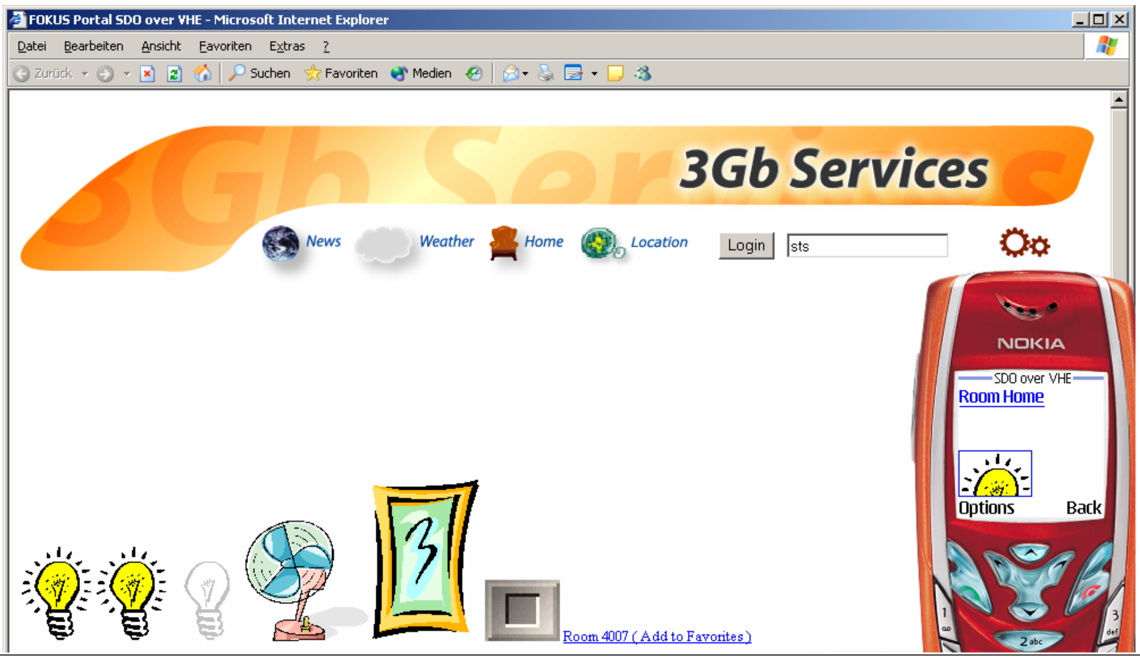

FOKUS Portal SDO over VHE - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Medien
3Gb Services
News
Weather
Home
Location
Login
sts
NOKIA
SDO over VHE
Room Home
Options
Back
Room 4007 (Add to Favorites)

Figure 4-21: Screenshot of the SDO Service depicting the status of a number of SDOs

The elements of the SDO system also provide the ambient information, which is stored in the Ambient Server and which is provided to other services through the I-centric User Interaction portal platform. In this way, the user can be notified by the SDO service if certain situations appear. For example, if lights are still switched on, although nobody is in the room. If necessary, the user can react to such messages by accessing the SDO service to control the relevant devices. Such mechanism is necessary for all situations in which the smart services cannot come to a decision and need therefore the user's assistance.

This service uses the personalization feature of the I-centric User Interaction portal platform to store the personal preferences of the users. For example, filtering rules, personalized names of devices, preferred and relevant locations, as well as the notification conditions are stored as preferences data. Furthermore, the service makes use of available location data. This means, when the user accesses the SDO service, all SDO elements are presented according to the location in which the user currently resides. In this way, the user does not to specify the current location of interest when the user is moving with a mobile terminal, such as a PDA with wireless LAN connection, through the different locations. The SDO elements to be presented are automatically selected based on the location information.

4.16 Summary

This chapter introduced a practical realization of an I-centric User Interaction system. This system implements the Service Adaptation Framework, which is defined in the approach to realize I-centric User Interaction as described in chapter 3. The several functional elements were implemented on different development and execution platforms. Based on the implementation, some exemplary services were developed in order to demonstrate the capabilities and the potential of the I-centric User Interaction approach. The service representing examples from different application domains provide the proof of concept.

The service developers do not need to consider the peculiarities of the Access Mechanism and used interface technologies on the client side. By applying the Generic User Interaction model, they can concentrate on the actual service logic leaving the presentation and the delivery to the I-centric User Interaction portal. The services also demonstrated that the use of the personalization function improves their behavior in terms of user-friendliness.

At time of writing this work, the development of the prototypical realization was still ongoing. The state of implementation was limited to a selected subset of relevant user interface technologies, i.e. WWW, WAP, VoiceXML, and SMS. However, due to technical restrictions not all relevant technologies could be implemented and tested, e.g. MMS because there were no according service centers provided by the mobile telecommunication network providers.

Similarly, the gathering of the terminal and network capabilities had to be simulated partially, because there were no real devices with a suitable support on the market or the information, provided by the vendors, was not sufficient respectively. In addition to the technical improvement and extension of further adaptor gateways, the current and future activities concentrate on new innovative concepts, such as multi-modal user interaction and advanced content adaptation algorithms like ‘summarizing’ functions. Likewise, the aspect of considering ambient information is not yet fully exhausted due to the limited availability of sensing devices. Because of the cooperation with related research projects, a better integration is expected in the further development.

However, the first results, as presented in this work, have already demonstrated the feasibility of the developed approach and the full potential of the vision. Whereas, the theoretical models concentrate only on the general concepts, the practical realization revealed some performance issues. The Service Adaptation pipeline accessing the Preferences Management System and the Ambient Information System is very complex resulting in a high demand on computing power. The implementation is still subject to optimization and other execution platforms have to be evaluated, likewise.

CHAPTER 5 Summary and Outlook

This chapter concludes this thesis. It summarizes the basic ideas and results of the developed approach. Furthermore, an outlook into the future development identifying some open issues is given.

5.1 Summary

Embedded into the concept of I-centric Communications, this work has presented an approach to realize I-centric User Interaction. The general vision of I-centric Communications defines a *user-centered approach* for the realization of services and applications. This requires to start analyzing the individual’s demand and to design a system with the necessary features to provide an intelligent behavior supporting the individual. Instead of just providing technology focused solutions without any adaptation to individuals, an I-centric system should provide services hiding technical details and considering the individual’s preferences as well as the individual’s environment. Individuals of any kind, at any life stage, with different technical competences and varying goals are supported in a user-friendly and comfortable way. This new user-centered design and development approach allows a new quality of services.

The vision of I-centric Communications identifies two different aspects in the service provisioning. The first aspect describes the *implicit service usage*. The service acts according to the user’s situation and preferences. There is no direct interaction between a user and the service. The interaction consists of the individual’s *actions*, such as walking, sleeping, etc., and the according *reactions* of the I-centric services, such as controlling devices and the environment. The second aspect describes the *explicit interaction* of the user with an I-centric service. This means, the user *communicates* with the service in a dialog. This aspect is denoted as *I-centric User Interaction* and is the subject of this work.

Corresponding to the vision of I-centric Communications, the adaptive, personalized, and ambient aware behavior of services is a required characteristic. Beside the actual service execution, these characteristics apply also to the interaction of the service with the user. This means, I-centric User Interaction signifies a personalized and ambient aware interaction between user and service. In addition, the users shall be able to interact with their services at any time and any place, which leads to the requirement of a *device-independent user interaction*. Device independence means that services do not have to deal with the peculiarities and individual capabilities of terminals and access networks. There are different user interface technologies, suitable for different situations and applications. According to the vision, the user interaction should not be limited to specific user interface technologies. Instead, the I-centric User Interaction should apply the most suitable technologies according to the situation and preferences of the user. This requires a *Generic User Interaction* approach, which is independent from specific user interface technologies.

The approach to realize I-centric User Interaction, presented in this thesis, identifies three areas of concern. Accordingly, this approach specifies a model for *Personalization*, a model for A*mbient Awareness*, and a model for *Generic User Interaction*. Each model defines suitable solutions and concepts for the respective area of concern. The models are interconnected defining necessary interfaces to exchange relevant information. In order to fulfill the objectives of I-centric User Interaction, all three models have to be realized coherently. This is achieved by the *Service Adaptation Framework*, which identifies all necessary functional elements based on the concepts, which are provided by the individual models.

The **Service Adaptation Framework** depicts a platform providing suitable interfaces to the services and defining the internal cooperation of the functional elements. The *Service Adaptation Function*, implementing the complete transformation pipeline, represents the central component. In its execution, this component considers the *Delivery Context* data provided by the Delivery Context Handler.

The Delivery Context Handler interrogates the Capability Manager, the Ambient Information System, and the Preferences Management System to collect all necessary data, needed to describe a Delivery Context. The Capability Manager captures the capabilities of the Access Mechanism, including the capabilities of the terminal and of the access network. The Ambient Information System provides the relevant ambient information, i.e. the ‘AmbientProfile’, and the Preferences Management System provides the preference settings from the currently effective User Profile.

Furthermore, the Service Adaptation Framework defines a suitable *session concept*, which enables the suspension and resumption of the service usage. The session concept differentiates between *Access Sessions* and *Service Sessions*. In this way, the user can interrupt the service usage at any time and can continue it later at the same stage, possibly using a different terminal with a different user interface technology.

The **Personalization** model defines an information model to structure and to describe personal preferences. The user’s preference settings are considered in the service execution, i.e. by the service, and in the realization of the interaction between user and service, i.e. by the Service Adaptation Function. Therefore, the relevant user preferences are included in the Delivery Context description. Furthermore, the Personalization model defines necessary functions and derives suitable interfaces, which are provided to internal and external components.

The information model of the proposed personalization concept defines the actual preference data as Attributes consisting of name value pairs. There are different categories of preferences: *general user preferences* and *service specific preferences*. The preference settings are grouped to User Profiles. Each user can have multiple User Profiles for different purposes. In order to select a User Profile, the Selection Context of each User Profile is evaluated, which is to be specified by the user. The Selection Context specifies the conditions for the selection of a certain User Profile. For example, it can contain timely conditions as well dependencies on locations and terminals. Furthermore, the user can explicitly select a specific User Profile omitting the automatic selection. In this way, the users can flexibly control and configure their services to behave personalized.

Based on the relevant usage scenarios, the Personalization model specifies necessary and elementary function to manage and access preference data. It defines an *internal* and an *external interface* to be provided to internal and external components respectively. The interfaces differ in access rights and functional extend to ensure the protection of personal data and to avoid any kind of misusage.

The **Ambient Awareness** model specifies a concept to gather and manage ambient information. This information can be provided by according sensor networks, which can sense physical data of the environment. These ‘raw’ data do not necessarily have a reasonable meaning suitable for the interpretation by the Service Adaptation Function and the services. Therefore, this model applies interpreter functions, which are able to revaluate the raw data to a higher semantic representation, suitable for further processing.

Furthermore, the Ambient Awareness model defines a mechanism to describe ambient information. This mechanism defining a corresponding 'AmbientInformation' module depicts an appropriate extension of the Delivery Context. In this way, relevant ambient information can be included in the Delivery Context and can be provided to the Service Adaptation Function as well as to the services. The ambient information is managed by the Ambient Information Server, which provides interfaces to store and query ambient information. The sources store plain data records, consisting of the actual ambient data and a location reference, to the server. Based on the location references, the server can compile an 'AmbientProfile', which contains all ambient information according to a specific location area and relevant to the service usage.

The **Generic User Interaction** model separates the user interface from the actual service logic. This means, the service developers have only to provide a generic description of the user interaction. They do not have to implement a support for specific user interface technologies directly. Because, the user interaction description, as proposed in this thesis, is completely independent from specific user interfaces technologies, the approach is denoted as *Generic User Interaction*. The model specifies an according language to describe user interaction in a generic way – the *Generic User Interaction Markup Language* (GUIML).

GUIML is based on standard W3C technologies, namely XHTML, XForms, and SMIL. Therefore, it is an XML-based language, which can be translated to specific user interface technologies by applying suitable transformation rules. The transformation is carried out in a pipeline by the *Service Adaptation Function*. The pipeline defines all individual steps, which are necessary to create a suitable representation of the user interface in required user interface markup languages, such as HTML or WML. In order to realize the service interaction in a personalized and ambient aware way, the individual steps of the transformation pipeline consider relevant personal preference data and ambient information, which are provided by the Delivery Context.

The Delivery Context describes all data relevant for the service execution and for the interaction with the user, i.e. it contains the capabilities of the Access Mechanism, user preferences, and ambient information. The Generic User Interaction model introduced a language for the description of the Delivery Context, namely the *eXtensible Delivery Context Language* (XDCL). The data of the Delivery Context are collected from different sources, such as the Capability Manager, the Preferences Management System, and the Ambient Information System. They are exchanged and provided to the Service Adaptation Function and the services using XDLC.

XDCL is an XML-based language. It defines the general structure of the Delivery Context and uses modules for specific extensions. That is why it is called extensible. In this way, this language does not prescribe concrete type of context data, which cannot be defined entirely and commonly because of the heterogeneity of relevant information. For each specific purpose, suitable modules have to be defined, which then can be used in the Delivery Context. If a module is unknown to a component, which interprets a Delivery Context, it has just to ignore the respective data. This depicts an open and flexible approach, which is capable to reflect specific requirements of various application scenarios.

The advantages of the presented I-centric User Interaction approach can be summarized to the support the users in the interaction with their services at any time, at any location, with any terminal and user interface technology, in consideration of their individual preferences and of their current environment. The developed and presented concepts realize these features providing a suitable support for the services. Using the proposed Service Adaptation Framework, the effort for service developers is kept small, although the experienced quality and benefits for the users are increased clearly. Of course, the automatic adaptation does not and will not completely replace existing approaches, in which specific user interface technologies are directly supported by the services. According to the type of an application, the most suitable and efficient approach has to be selected respectively.

5.2 Outlook

The presented approach to realize I-centric User Interaction fulfils the identified objectives. The prototypical realization of the Service Adaptation Framework and the exemplary services demonstrate the feasibility of the developed concepts. However, the currently existing prototype system uses only a selection of diverse user interface technologies demonstrating different alternatives, such as speech-based, message-based, as well as WWW- and WAP-based interaction. In the future development, the support for additional user interface technologies should be integrated likewise, in order to continue examining the applicability of the developed approach.

Similarly, the services, which have been selected for demonstration and evaluation purposes, utilize the different provided features successfully. However, this kind of evaluation should be continued, preferably with external service developers and service providers in order to obtain reasonable feedback and assessments. This feedback should reveal future requirements to be fulfilled and existing weak points to be resolved.

During the development of the individual concepts and of the prototype system, already some open issues and additional required functional have been identified. For example, service developer needs to learn a new language to describe the user interaction. Although, the proposed concept is also based on XML, like existing approaches, such as HTML and WML, it is not trivial to understand and to have a complete command of this language immediately. Additionally, the automatic generation of the user interfaces requires some experience with this approach. This requires a suitable and efficient service development support, which allows an immediate evaluation of the results. The future activities should include the development of a tool supporting the service developer in the design of user interfaces and in the testing of the resulting user interaction.

Furthermore, the approach should be investigated from a business viewpoint. The question to be answered is how the different resource usage can be accounted. When the service sends a message to the user, different alternatives, which vary in quality but also in costs, can be chosen by the systems. For example, sending an e-mail is usually cheaper then sending a multimedia message (MMS). In order to provide efficient and profitable business and revenue models, suitable accounting algorithms are required, which reflect the complexity of the underlying technologies and respective cost models. On the one hand, this requires that the individual functions of the portal platform support a suitable accounting mechanism by providing reasonable usage data in a common format. On the other hand, according price models have to be defined, which allow charging the costs to the respective users. The accounting system should be transparent to the users and support innovative business models, such as prepaid, advertisements-based, and coupon-based application scenarios.

As already identified in the presentation of the individual concepts, a special attention has to be put on security and privacy issues. The personal data, such as the user’s preferences but also ambient information, revealing details about the individual, have to be secured against misuse. Any kind of potential misuse immediately threatens the trust of the user and in turn the successful operation of the portal platform and the services. Future activities should carefully analyze all possible security threads and provide according solution approaches.

A further aspect, with regard to the acceptance of the proposed approach by the users, deals with the issue of transparency. The system and the service behavior are dynamically controlled by a number of factors. Because of the potential complexity, the user might not completely understand the behavior, which could lead to dissatisfaction. Before putting the service platform as well as individual services in commercial operation, careful and extensive user studies should be carried out first. These should give the necessary indications, to which degree the complex functions including autonomous decisions, reasoning, and self-learning behavior can be utilized. The complete system has to ensure that the users can always understand the system behavior and feel being rather supported than misunderstood and bothered. This includes also that users have at any time the full control over the behavior of their services. This aspect is an integrative part of user-centered design approach. Any kind of negative experience would hinder the success of the any service and hence of the portal platform.

The presented approach to realize I-centric User Interaction applies and reuses existing standards and concepts at best means. However, the corresponding research and the standardization activities in the different areas of concern have not been finalized completely or even have not yet provided suitable results respectively. To different extents, these areas comprise personalization, ambient awareness, and the device independent user interaction, including the capturing and provisioning of capabilities. The future work should carefully follow these activities, trying to adopt arising standards and, likewise, to influence ongoing standardization processes based on the concepts, which were developed in the presented approach.

As the experiences of the prototypical realization have demonstrated, the implementation of the individual concepts requires a careful selection of efficient and capable execution platforms and operating systems. Future activities should determine the bottlenecks in the execution of the transformation pipeline and develop according solutions respectively. The current prototypical implementation has been focused to provide an evaluation and demonstration platform solely and have been adapted repeatedly over the time. In order to obtain a more efficient implementation, some components require a complete and clean re-implementation following a clear software design.

Although the approach of I-centric Communications requires the coherent application of the developed models, some integrative concepts can be extracted and pursed separately. Whereas the complete package does not have to be suitable for all kind of application scenarios, partial solutions approaches could be sufficient and adequate, likewise. Therefore, the future activities should not insist on an interconnected approach, but should continue pursuing open and flexible concepts for the individual areas of concern.

CHAPTER 6 References

6.1 Literature

[3GPP-TS22.121] 3GPP TS 22.121: *Digital cellular telecommunications system (Phase 2+) (GRM); Universal Mobile Telecommunications System (UMTS); Service aspects; The Virtual Home Environment*. Version 5.3.1, June 2002.

[3GPP-TS22.240] 3GPP TS 22.240: *3GPP Generic User Profile (GUP)*, Version 1.0.0, September 2002.

[3GPP-TS23.241] 3GPP TS 23.241: *3GPP Generic User Profile – Data Description Framework*, Version 0.3.0, February 2002.

[3GPP-TS26.234] 3GPP TS 26.234: Technical Specification Group Services and System Aspects: *Transparent end-to-end PSS, protocols and codecs*. Version 1.5.1, March 2001

[APPEL] W3C: *A P3P Preference Exchange Language 1.0 (APPEL1.0)*. W3C Working Draft, 15 April 2002

[ARB] Arbanowski, St; Breugst, M; Busse, I; Magedanz, T.: *Impact of Standard Mobile Agent Technology on Telecommunications*. 5th Conference on Computer Communications, AFRICOM-CCDC'98, Tunis, Tunisia, October 20-22, 1998, pp. 189-203

[ARBMEER1] van der Meer, S.; Arbanowski, S.; Magedanz, T.: *An Approach for a 4th Generation Messaging System*. Proceedings of The Fourth International Symposium on Autonomous Decentralized Systems, ISADS'99, Tokyo, 1999

[ARBMEER2] van der Meer, S.; Arbanowski, S.: *Service Personalization for Unified Messaging Systems*. Department for Open Communication Systems (OKS), Technical University Berlin

[BALL] Ball, T.; Colby, C.; Danielsen, P.; Jategaonkar, L.; Jagadeesan, R.; Läufer, K.; Mataga, P., Rehor, K.: *Sisl: Several Interfaces, Single Logic*. International Journal of Speech Technology, Volume 3, Issue 2, pp. 93-108, June 2000

[BARBIR] Barbir, K.; Bennett, N.; Penno, R.; Pham, H. T.; et al: *A Framework for Service Personalization*, Internet Draft, June 2002.

[BEKKUM] van Bekkum, M.; Bijlsma, M.; van Kranenburg, H.; Lankhorst, M.: *Personal Service Environment: Analysis and Research Issues*. Telematica Instituut, GigaMobile/D3.2.4, December 15, 2000

[BUNT] Bunt, H.; Ahn R.; Beun R.; Borghuis & Kees van Overveld T.: *Cooperative Multimodal Communication in the DenK Project*. Institute for Language Technology and Artificial Intelligence (ITK), Tilburg University, Tilburg, The Netherlands; Institute for Perception Research (IPO), Eindhoven, The Netherlands; Faculty of Mathematics and Computing Science Eindhoven University of Technology, Eindhoven, The Netherlands

[BUTLER] Butler, M. H.: *Current Technologies for Device Independence*. HP Laboratories Bristol, 2001

[BUTLER1] Butler, M. H.: *Some Questions and Answers On CC/PP and UAProf*. Publishing Systems and Solutions Laboratory, HP Laboratories Bristol

[BUTLER2] Butler, M. H.: *Using capability classes to classify and match CC/PP and UAProf profiles*. Publishing Systems and Solutions Laboratory, HP Laboratories Bristol

[BUTLER3] Butler, M. H.: *Implementing Content Negotiation using CC/PP and WAP UAProf*. Information Infrastructure Laboratory, HP Laboratories Bristol; HPL-2001-190; August 7, 2001

[CARROLL] Carrol, L.; Peake, M.; Dickerman, C. (Ed.): *Alice's Adventure in Wonderland / Through the Looking-Glass*. Bloomsbury USA, October 2001, ISBN: 1582342229

[CHEEPEN1] Cheepen, C.: *Dialog Definitions and Discoursal Models for Advanced Voice Dialogs – Working Paper 1*, Department of Sociology, University of Surrey, Guildford, 1996

[CHEEPEN2] Cheepen, C.: *Categories of Dialog for Advanced Voice Dialogs – Working Paper 2*. Department of Sociology, University of Surrey, Guildford, 1996

[COLE] Cole, R.: *Spoken Output Technology – Survey of the State of the Art in Human Language Technology*. Center for Spoken Language Understanding, Oregon Graduate Institute of Science & Technology, Portland, Oregon, USA, 1995

[COLEZUE] Cole, R.; Zue, V.: *Spoken Language Input – Survey of the State of the Art in Human Language Technology*. Center for Spoken Language Understanding, Oregon Graduate Institute of Science & Technology, Portland, Oregon, USA, 1995

[CONVIGO] Convigo Inc.: *Convigo Platform: Technology White Paper*. Convigo Inc., 2001

[CORBA] OMG: *The Common Object Request Broker – Architecture and Specification*. Object Management Group, Revision 2.2, 1998

[DECUMS] Digital Equipment Corporation: *Universal Messaging Whitepaper*. May 1997

[DENWAI] Denecke, M.; Waibel, A.: *Dialog Strategies Guiding Users to their Communicative Goals.* Interactive Systems Labs., Carnegie Mellon University, Pittsburgh, USA

[DEY] Dey, A.; Abowd, G.: *Towards a Better Understanding of Context and Context-Awareness.* Proceedings of the Computer-Human Interaction 2000 (CHI 2000), Workshop on The What, Who, Where, When, and How of Context-Awareness, April 2000.

[ECKARDT] Eckardt, T.; Magedanz, T.; Pfeifer, T.: *On the Convergence of Distributed Computing and Telecommunications in the Field of Personal Communications*. Franke, K. et al. (Ed.).: Proc. of Kommunikation in Verteilten Systemen, pp. 46-60, KiVS'95, Chemnitz, Feb. 1995, Berlin: Springer, 1995

[ECKARDT1] Eckardt, T.; Magedanz, T.; Ulbricht, C.; Popescu-Zeletin, R.: *Generic Personal Communications Support for Open Service Environments*. Proc. IFIP World Conference on Mobile Communications, Canberra, Australia, Sep. 1996

[ECKARDT2] Eckardt, T. (Ed.): *Deutsche Telekom Project "Personal Communications Support in TINA", Report No. 1*. GMD FOKUS, June 1996

[ETSITS22.01] ETSI Technical Specification TS 22.01v3.1.0: *Universal Mobile Telecommunication Systems (UMTS). Service Aspects. Service Principles.* Sophia Antipolis, France, 1997

[ETSITS22.70] ETSI Draft 22.70 v.0.0.3: *Virtual Home Environments*. Sophia Antipolis, France, 1997

[FAROOGUI] Faroogui, K.; Logrippo, L.: *Introduction to ODP Computational Model.* Department of Computer Science, University of Ottawa, Canada

[FINK] Fink, J.; Koenemann, J.; Noller, S.; Schwab, I.: *Putting Personalization into Practice*. Communications of the ACM, Vol. 45, No. 5, pp. 41-42, ACM Press, May 2002

[FLAMMIA] Flammia, G.: Discourse Segmentation of Spoken Dialog: *An Empirical Approach. Laurea.* Universita di Roma, La Sapienza, Italy, 1988

[FURUI1] Furui, S.: *Speaker-independent isolated word recognition using dynamic feature of the speech spectrum*. IEEE Transactions on Acoustics, Speech and Signal Processing, 29(1):59-59, 1986

[FURUI2] Furui, S.: *Digital Speech Processing, Synthesis, and Recognition*. Marcel Dekker, New York, 1989

[GAMMA] Gamma, E.; Helm, R. ; Johnson, R.; Vlissidess, J.: *Design Patterns: Elements of Reusable Object-Oriented Software,* Addison Wesley, 1995

[GMPCSMOU] ITU-T: *Memorandum of Understanding on Global Mobile Personal Communications by Satellite*. World Telecommunications Policy Forum, October 1996 / February 1997

[GOEBEL1] Göbel, S.; Buchholz, S.; Ziegert, T.; Schill, A.: *Device Independent Representation of Web-based Dialogs and Contents*. Proc. of the IEEE Youth Forum in Computer Science and Engineering (YUFORIC'01), Valencia, Spain, Nov 29-30, 2001

[GOEBEL2] Göbel, S.; Buchholz, S.; Ziegert, T.; Schill, A.: *Software Architecture for the Adaptation of Dialogs and Contents to Different Devices*. TU Dresden, Department of Computer Science, 2002

[GPS] Poizner, S.; Todd, K.: *Extending GPS capabilities*. Wireless Review, 16(1999)9, Overland Park, KS, 1 May 1999

[GUNTERMANN] Guntermann, M.; et al: *Integration of Advanced Communication Services in the Personal Services Communication Space – A Realisation Study*. Proc. of the RACE Intl. Conference on Intelligence in Broadband Service and Networks (IS&N) 1993 (Mobilise), pp. II/1/p.1-12

[HENNICKER] Hennicker, R.; Koch, N.: *A UML-based Methodology for Hypermedia Design*. UML'2000 – The Unified Modeling Language – Advancing the Standard, volume 1939 of Lecture Notes in Computer Science, York, England, Springer Verlag, October 2000

[HERMANSKY] Hermansky, H.: *Perceptual linear predictive (PLP) analysis for speech*. Journal of the Acoustical Society of America, 87(4):1738-1752, 1990

[HÜNERBERG] Hünerberg, J.: *Evaluating Object oriented Databases for a Location Aware Service and Application Platform*. Diploma Thesis, Technical University Berlin, Berlin, December 1999

[HUNTER] Jane Hunter: *Adding Multimedia to the Semantic Web Building on MPEG-7 Ontology*. Proceedings of the First Semantic Web Working Symposium (SWWS), Stanford, USA, 2001

[ICS1] van der Meer, S.; Arbanowski, St.; Steglich, St.; Popescu-Zeletin, R.: *The Human Communication Space Towards I-centric Communications.* Workshop: The What, Who, Where, When, Why and How of Context-Awareness, ACM Conference on Human Factors in Computing Systems, CHI2000, The Hague, The Netherlands, April 1-6, 2000

[ICS2] Arbanowski, St.; van der Meer, S.; Steglich, St.; Popescu-Zeletin, R.: *I-centric Communications. Informatik - Forschung und Entwicklung*. Organ des Fachbereichs 2 der Gesellschaft für Informatik e.V. (GI), Springer-Verlag, Band 16, Heft 4, S. 225-232, 2001, ISSN 0178-3564 IFENEI

[ICS3] van der Meer, S; Arbanowski, St; Steglich, St: *Flexible Control of Media Gateways for Service Adaptation.* Proc of the 5th IEEE Intelligent Network Workshop, Cap Town, South Africa, May 07-11, 2000, ISBN 0-7803-6317-5

[ICS4] Arbanowski, St; van der Meer, S: Popescu-Zeletin, R: *I-centric Services in the Area of Telecommunication 'The I-Talk Service'*. Proc. of the 6th IFIP TC6/WG6.7 Conference on Intelligence in Networks, SmartNet 2000, Vienna, Austria, September 18-22, 2000, pp. 499-508, ISBN 0-7923-7932-2

[ICS5] Arbanowski, St.; van der Meer, S.; Steglich, St.; Popescu-Zeletin, R.: *The Human Communication Space: Towards I-centric Communications.* Volume 5, Personal and Ubiquitous Computing, Issue 1, pp. 34-37, ISSN 1617-4909

[ICS6] Steglich, St.; Popescu-Zeletin, R.: *Towards I-centric User Interaction*. ICME 2001 International Conference on Multimedia and Expo, ICME 2001, Tokyo, Japan, August 22-25, 2001, ISBN 0-7695-1198-8

[ICS7] van der Meer, S; Arbanowski, St; Steglich, St: *User-Centric Communications*. Proc. of the IEEE ICT 2001 – IEEE International Conference on Telecommunications, Romania, 2001, Volume 4, pp. 452-444, ISBN 973-99995-3-0

[ICS8] van der Meer, S., Steglich, St., Arbanowski, St.: *From Unified Messaging towards I-centric - Services for the Virtual Home Environment*. Proc. of the 6th IEEE Intelligent Network Workshop, IN2001, Boston, MA, USA, May 6-9, 2001, ISBN 0-7803-7047-3

[ICS9] Arbanowski, St., Steglich, S.: *Profiling Contexual Information*. IEEE International Conference on Parallel Architectures and Compiling Techniques – Proc. of the Workshop on Ubiquitous Computing and Communication, Barcelona, 2001

[ICS10] Arbanowski, St.; Steglich, S.: *Profile Information based Service Creation*. Proc. of the 4th Asia-Pacific Symposium on Information and Telecommunication Technologies, Tribhuvan University, Kathmandu, Nepal, 2001

[ICS11] Arbanowski, St., Steglich, St.: *Service Architectures for 3G and Beyond*. SICE Annual Conference 2003, Fukui, Japan, Augut 4-6, 2003

[ICS12] Radusch, I., Arbanowski, St., Steglich, St., Popescu-Zeletin, R.: *I-Centric Services based on Super Distributed Objects*. Med-Hoc NET 2003 Workshop, Mahdia, Tunesia, March 26-27, 2003

[ICS13] Steglich, St.; Vaidya, R. N.; Gimpeliovskaja, O.; Arbanowski, St.; Popescu-Zeletin, R.; Sameshima, Sh.; Kawano, K.: *I-Centric Services Based on Super Distributed Objects*. ISADS 2003, Proc. of the 6th International Symposium on Autonomous Decentralized Systems, Pisa, Italy, April 9-11, 2003, pp. 232-239, ISBN 0-7695-1876-1

[ICS14] Arbanowski, St.; Steglich, St.; Popescu-Zeletin, R.: *Super Distributed Objects - an execution environment for I-centric Services*. Proc. of the 9-th IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS 2003F), Capri Island, Italy, October, 1-3, 2003

[ICS15] van der Meer, S; Arbanowski, St: *Service Interoperability through advanced Media Gate-ways.* Proc. of the 6th IFIP TC6/WG6.7 Conference on Intelligence in Networks, SmartNet 2000, Vienna, Austria, September 18-22, 2000, pp. 583-595, ISBN 0-7923-7932-2

[ICS16] van der Meer, S; Arbanowski, St: *Flexible Media and Content Adaptation for Communica-tion Systems.* Proc. of the IEEE Conference on Protocols for Multimedia Systems, PROMS 2000, Cracow, Poland, October 22-25, 2000, pp. 461-477, ISBN 83-88309-05-6

[IMODE] Frengle, N.: *i-Mode: A Primer*. Wiley, John & Sons, Incorporated, ISBN**:** 0764548840, January 2002

[ISOMPEG21] International Organisation For Standardisation (ISO/IEC): *MPEG-21 Overview*.

[ISOMPEG7]	Martínez, J. M. (UPM-GTI, ES): *MPEG-7 Overview*. International Organisation For Standardisation (ISO/IEC) JTC1/SC29/WG11 N3445, http://www.cselt.it/mpeg/standards/mpeg-7/mpeg-7.htm, July 2002
[ISTAG]	Ducatel, K.; Bogdanowicz, M.; Scapolo, F.; Leijten, J.; Burgelman, J-C.: *Scenarios for Ambient Intelligence in 2010*. Final Report, IPTS-Seville, Feb 2001, EC 2001
[ITU-Q1711]	International Telecommunication Union: *ITU-T Q.1711 – Network functional model for IMT-2000*, ITU-T, March 1999.
[JASPER]	Jasper, K.: *Management of Network and Terminal Capabilities for VHE Service Adaptation*. Diploma Thesis, Technical University Berlin, Berlin, March 2003
[JURMAR]	Jurafsky, D.; Martin, J. H.: *Speech and Language Processing – An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Draft of June 8, 1999, Prentice Hall, Inc, Englewood Cliffs, New Jersey 07632, 1999
[KAASINEN]	Kaasinen, E.; Aaltonen, M.; Kolari, J.; Melakoski, S.; Laakko2, T.: *Two Approaches to Bringing Internet Services to WAP Devices.* The Ninth International World Wide Web Conference, 2000
[KLATT]	Klatt, D. H.: *Review of text-to-speech conversion for English*. Journal of the Acoustical Society of America, 82(3), pages 737-793, 1987
[KOBSA]	Kobsa, A.; Koenemann, J.; Pohl, W.: *Personalized hypermedia presentation techniques for improving online customer relationships.* The Knowledge Enginerring Review, Vol. 16:2, Cambridge University Press, 2001, pp. 111-155.
[KOCH]	Koch, N.; Kraus, A.: *The Expressive Power of UML-based Web Engineering.* Second International Workshop on Web-oriented Software Technology (IWWOST´02). CYTED, June 2002.
[LANGHAM]	Langham, M.: *Cocoon: Building XML Applications*. New Riders Publishing, 2002
[LEMLOUMA]	Lemlouma, T.; Layaida, N.: *The Negotiation of Multimedia Content Services in Heterogeneous Environments*. OPERA Project, 2001
[LIBERTYA]	Liberty Alliance Project: *Liberty Architecture Overview*, November 2002.
[MAGEDANZ]	Magedanz, T. (Ed.): *BERKOM II Project “IN/TMN Integration”, Deliverable 7: Personal Communication Support System: Specification of Profiles, System Components, and Applications*. Technical University of Berlin, Institute for Open Communications System (OKS), November 1995
[MANBER]	Manber, U.; Patel, A.; Robison, J.: *Experience with Personalization on Yahoo!*, in: *Communications of the ACM*, Vol. 43, No. 8, ACM Press, August 2000, pp. 35-39.
[MANDYAM]	Mandyam, S.; Vedati, K.; Kuo, C., Wang, W.: *User interface Adaptations: Indispensable for Single Authoring*. Position Paper of W3C Workshop on Device Independent Authoring Techniques, SAP University, St. Leon-Rot, Germany, September 25-26, 2002
[MENKHAUS]	Menkhaus, G.: *Architecture for Client-Independent Web-based applications.* IEEE Proceedings of the TOOLS-Europe Conference, Zürich, March 12-14, 2001
[MOHAMAD]	Mohamad, Y., Carlos, A. Velasco: *Supporting Device Independent Accessible Authoring by a Next Generation Web Publishing Framework*. Position Paper of W3C Workshop on Device Independent Authoring Techniques, SAP University, St. Leon-Rot, Germany, September 25-26, 2002
[MOHR]	Mohr, W.: *The Wireless World Research Forum (WWRF) Towards Systems Beyond 3G*. The Proceedings of the Korean Institute of Communication Sciences, vol. 19, No. 7, July 2002, pp. 56, invited paper.
[MROHS]	Mrohs, B.: *Service Adaptation Framework for Heterogeneous Terminal Devices*. Diploma Thesis, Technical University Berlin, Berlin, January 2003
[MSPASSPORT]	Microsoft Corporation: *Microsoft .net Passport Review Guide.* November 2002.
[MUELLER]	Müller, A.; Forbrig, P.; Cap, C.: *Model-Based user interface Design Using Markup Concepts.* DSV-IS, 2001
[NANNEMAN]	Nanneman, D.: *Unified Messaging: A Progress Report. - Telecommunications Magazine*. March 1997
[NITF]	NITF 3.0: *News Industry Text Format*. International Press Telecommunications Council (IPTC), Oct 11, 2001
[ODP]	Raymond, K.: *Reference Model of Open Distributed Processing (RM-ODP): Introduction*. - Center for Information Technology Research, University of Queensland, Australia; also in: Proc. of the International Conference on Open Distributed Processing, ICODP’95, Brisbane, Australia, pp. 20 – 24, February, 1995
[OMGMDA]	Siegel, J.; OMG Staff Strategy Group: *Developing in OMG’s Model-Driven Architecture*. White Paper, Revision 2.6, Object Management Group, OMG document omg/01-12-01, November, 2001

[P3P]	W3C: *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*. W3C Recommendation, 16 April 2002
[PFEIFER]	Pfeifer, T. (Ed.): *BERKOM Project "Intelligent Personal Communication Support System", Deliverable 0: Design criteria and preliminary description of functionality*. GMD FOKUS, December 1995
[PFEIFER1]	Pfeifer, T.: *Automatic Conversion of Communication Media*. Dissertation, Technical University Berlin, Berlin, Germany, 2000, ISBN 3-88457-374-8
[PFEIFER2]	Pfeifer, T; Arbanowski, St; Popescu-Zeletin, R.: *Resource Selection in Heterogeneous Communication Environments using the Teleservice Descriptor*. 4th COST 237 Workshop, Lisboa, Portugal, December 15-19, 1997, pp. 132-153, ISBN 3-540-63935-7
[PFEIFER3]	Pfeifer, T.; Popescu-Zeletin, R.: *A Modular Location-Aware Service and Application Platform*. The Fourth IEEE Symposium on Computers and Communications, ISCC'99, Red Sea, Egypt, July 6-8, 1999
[PIAVIDAD5.1]	Fraunhofer FOKUS: *Specification des Voice Portals*. BMBF PI-AVIda Projekt. Dezember 2002
[PIAVIDAD5.2]	Fraunhofer FOKUS: *Specification des Multimedia Portals*. BMBF PI-AVIda Projekt. Januar 2003
[PIAVIDAD5.3]	Fraunhofer FOKUS: *Prototyp des Voice Portals*. BMBF PI-AVIda Projekt. Juli 2003
[PIAVIDAD5.4]	Fraunhofer FOKUS: *Prototyp des Multimedia Portals*. BMBF PI-AVIda Projekt. Juli 2003
[PICONE]	Picone, J.: *Signal Modelling Techniques In Speech Recognition*. Texas Instruments Systems and Information Sciences Laboratory, Tsukuba Research and Development Center, Tsukuba, Japan, 1993
[POLIFRONI]	Polifroni, J.; Seneff, S.; Glass, J.; Hazen T.: *Evaluation Methodology for a Telephone-Based Conversational System*. Spoken Language Systems Group, Laboratory for Computer Science, MIT, Cambridge, Massachusetts, USA
[PRLY21MOB]	The Parlay Technical Team: *Parlay APIs 2.1, Mobility Data Definitions*. Issue 1.1, 26 June 2000
[RABJUANG]	Rabiner, L. R.; Juang, B. H.: *Hidden Markov Models for Speech Recognition – Strength and Limitations*. AT&T Bell Labs, Murray Hill, New Jersey
[RÄCK]	Räck, C.: *Profile and Preferences Management System*. Diploma Thesis, Technical University Berlin, Berlin, January 2003
[RAKOTONI]	Rakotonirainy, A.; Wai Loke, S.; Fitzpatrick, G.: *Context-Awareness for the Mobile Environment*. CHI2000 Workshop #11 Proposal, April 2000.
[RAYMOND]	Raymond, K.: *Reference Model of Open Distributed Processing (RM-ODP): Introduction*. International Conference on Open Distributed Processing, ICODP'95, Brisbane, Australia, 1995
[RDF]	Lassila, O.; Swick, R.: *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation, 22 February 1999
[RFC1766]	IETF: *Tags for the Identification of Languages*. IETF RFC-1766, March 1995
[RFC2293]	Internet Engineering Task Force (IETF): *Representing Tables and Subtrees in the X.500 Directory*. IETF RFC 2293, March 1998.
[RFC2396]	Berners-Lee, T.; Fielding, R.; Masinter, L.: *Uniform Resource Identifiers (URI): Generic Syntax*. Internet Engineering Task Force (IETF) RFC 2396, August 1998
[RFC2506]	Holtman, K.; Mutz, A.; Hardie, T.: *Media Feature Tag Registration Procedure*. RFC 2506. IETF Request for Comments: ftp://ftp.isi.edu/in-notes/rfc2506.txt
[RFC2531]	Klyne, G.; McIntyre, L.: *Content Feature Schema for Internet Fax*. RFC 2531, IETF Request for Comments: ftp://ftp.isi.edu/in-notes/rfc2531.txt
[RFC2533]	Klyne, G.: *A Syntax for Describing Media Feature Sets*. RFC 2533, IETF Request for Comments: ftp://ftp.isi.edu/in-notes/rfc2533.txt
[RFC2534]	Masinter, L.; Wing, D.; Mutz, A.; Holtman, K.: *Media Features for Display, Print, and Fax*. RFC 2534, IETF Request for Comments: ftp://ftp.isi.edu/in-notes/rfc2534.txt
[RFC2616]	IETF: *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC-2616, June 1999
[ROSSI]	Rossi, G.; Schwabe, D.; Guimarães, R.: *Designing Personalized Web Applications,* in International World Wide Web Conference: Proceedings of the tenth international conference on World Wide Web, ACM Press, 2001, pp. 275-284.
[SANDKUHL]	Sandkuhl, K.: *Ein Referenzmodell für informationslogistische Anwendungen.* in Deiters, W.; Lienemann, C.: *Report Informationslogistik.* 1. Auflage, Juli 2001
[SCHILIT]	Schilit, B.; Theimer, M.: *Disseminating Active Map Information to Mobile Hosts.* IEEE Network, Volume 8, Number 5, September/October 1994, pp. 22-32.
[SUNEJB]	Sun Microsystems, Inc: *Enterprise JavaBeans Specification*, Version 2.0, August 2001.

[SUNJAVASPEC]	Sun Microsystems, Inc: *The Java Language Specification*, Second Edition. Sun Microsystems, Inc., 2000
[SUNJMFGUIDE]	Sun Microsystems, Inc.: *Java Media Framework API Guide*. Sun Microsystems, Inc., November 19, 1999
[SUNJMFSPEC]	Sun Microsystems, Inc.: *Java Media Framework Specification 2.0*. Sun Microsystems, Inc., November 23, 1999
[SUNJSPSPEC]	Sun Microsystems, Inc: *Java Servlet API Specification 2.3*. September 2001.
[THOUNG]	Thuong, T. T.; Roisin, C.: *Structured Media for Authoring Multimedia Documents*. International Workshop on Web Document Analysis, Seattle, Washington, USA, September 8, 2001
[TINASA]	Kristiansen, L. (Ed.): *Service Architecture*. Version 5.0 Jun 1997, TINA-Consortium, http://www.tinac.com/specifications/documents/sa50-main.pdf
[UIML]	Harmonia, Inc.: *User Interface Markup Language (UIML) v3.0. First Draft Specification.*. Harmonia, Inc., February 12, 2002
[UML1]	Fowler, M.: *UML Distilled – Applying the Standard Object Modeling Language*. Addison-Wesley Object Technology Series, Addison-Wesley, Massachusetts, 1997
[UML2]	*UML Notation Guide*. Rational Software Corporation, www.rational.com/uml, 1997
[UMTSR1]	UMTS Forum Report 1: *A Regulatory Framework for UMTS*, June 1997.
[USZKOREIT]	Uszkoreit, H.: *Discourse and Dialog – Survey of the State of the Art in Human Language Technology*. Center for Spoken Language Understanding, Oregon Graduate Institute of Science & Technology, Portland, Oregon, USA, 1995
[UUID]	*Information technology - Open Systems Interconnection - Remote Procedure Call (RPC)*. February 2003
[VDMEER]	van der Meer, S.; Arbanowski, St.; Magedanz, T: *An Approach for a 4th Generation Messaging System*. Proc. of The Fourth Intl. Symposium on Autonomous Decentralized Systems, ISADS'99, Tokyo, March 21-23, 1999
[VEDATI]	Vedati, K.: Convigo: *Recommendations for Improving Device Independent Presentation Authoring*. Convigo, 2001
[VHE-UD]	Fraunhofer FOKUS, Fraunhofer ISST: *VHE-Umgebung und Dienste (VHE-UD) Meilenstein M3*. 2002.
[W3CCCPP]	World Wide Web Consortium: *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies*. W3C Working Draft, November 8, 2002
[W3CCCPPARCH]	Nilsson, M.; Hjelm, J.; Ohto, H.: Composite Capabilities/Preference Profiles: Requirements and Architecture. W3C Working Draft:, http://www.w3.org/TR/CCPP-ra/
[W3CCSS2SPEC]	World Wide Web Consortium: *Cascading Style Sheets, level 2*. CSS2 Specification. W3C Recommendation, May 12, 1998
[W3CDCODI]	World Wide Web Consortium: *Delivery Context Overview for Device Independence*. W3C Working Draft; 13 December 2002; http://www.w3.org/TR/2002/WD-di-dco-20021213/
[W3CDIAC]	World Wide Web Consortium: *Authoring Challenges for Device Independence*. W3C Working Draft, October 18, 2002
[W3CDIAS]	World Wide Web Consortium: *Authoring Scenarios for Device Independence*. W3C Informal Public Draft, July 29, 2002
[W3CDIPRINC]	World Wide Web Consortium: *Device Independence Principles*. W3C Working Draft, Sep 18, 2001
[W3CDOM]	World Wide Web Consortium: *Document Object Model (DOM) Level 1 Specification*, Version 1.0. W3C Recommendation, October 1, 1998
[W3CHTML]	World Wide Web Consortium: *HTML 4.01 Specification*. W3C Recommendation, December 24, 1999
[W3CRDFPRIM]	World Wide Web Consortium: *RDF Primer*, W3C Working Draft, November 2002.
[W3CRDFSYN]	World Wide Web Consortium: *RDF/XML Syntax Specification (Revised)*. W3C Working Draft, November 2002.
[W3CSMIL]	World Wide Web Consortium: *Synchronized Multimedia Integration Language (SMIL 2.0)*. W3C Recommendation, August 7, 2001
[W3CSMIL1]	World Wide Web Consortium: *Synchronized Multimedia Integration Language (SMIL 1.0)*. W3C Recommendation, June 15, 1998
[W3CVXML]	World Wide Web Consortium: *Voice Extensible Markup Language (VoiceXML) Version 2.0*. W3C Working Draft, October 23, 2001
[W3CWAI]	World Wide Web Consortium: *XML Accessibility Guidelines*. W3C Working Draft, October 1, 2002
[W3CWCAG]	World Wide Web Consortium: *Web Content Accessibility Guidelines 1.0*. W3C Recommendation, May 5, 1999

[W3CXFORMS]	World Wide Web Consortium: *XForms 1.0*. W3C Candidate Recommendation, November 12, 2002
[W3CXHTML]	World Wide Web Consortium: *XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)*. W3C Recommendation, August 1, 2002
[W3CXHTMLB]	World Wide Web Consortium: *XHTML Basic*. W3C Recommendation, December 19, 2000
[W3CXHTMLS]	World Wide Web Consortium: *XHTML+SMIL Profile*. W3C Note, January 31, 2002
[W3CXHTMMO]	World Wide Web Consortium: *Modularization of XHTML*. W3C Recommendation, April 10, 2001
[W3CXML]	Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E.: *Extensible Markup Language (XML) 1.0 (Second Edition).* W3C Recommendation, World Wide Web Consortium, http://www.w3.org/TR/REC-xml, October 6, 2000
[W3CXMLSCH0]	Fallside, D. C.: *XML Schema Part 0: Primer*. W3C Recommendation, World Wide Web Consortium, http://www.w3.org/TR/xmlschema-0/, May 2, 2001
[W3CXMLSCH1]	Thompson, H. S.; Beech, D.; Maloney, M.; Mendelsohn, N.: *XML Schema Part 1: Structures*. W3C Recommendation, World Wide Web Consortium, http://www.w3.org/TR/xmlschema-1/, May 2, 2001
[W3CXMLSCH2]	Biron, P. V.; Malhotra, A.: *XML Schema Part 2: Datatypes*, W3C Recommendation, World Wide Web Consortium, http://www.w3.org/TR/xmlschema-2/, May 2001.
[W3CXSLT]	World Wide Web Consortium: *XSL Transformations (XSLT).* Version 1.0. W3C Recommendation, November 16, 1999
[WAGUAPRSP]	Wireless Application Group: *User Agent Profile Specification*, November 1999.
[WAPARCHSP]	WAP Forum: *Wireless Application Protocol Architecture Specification.*. WAP-210-WAPArch-20010712. WAP Forum, July 12, 2001
[WAPCSSSP]	WAP Forum: *WAP CSS Specification, Wireless Application Protocol*. WAP-239-WCSS-20011026-a. WAP Forum, October 26, 2001
[WAPMMSARC]	WAP Forum: *WAP MMS Architecture Overview*. WAP Forum, April 25, 2001
[WAPPUSH]	WAP Forum/Open Mobile Alliance: *WAP Push Architectural Overview.* WAP-250-PushArchOverview-20010703-a, July 2001, http:// www.wapforum.org
[WAPWML12]	WAP Forum: *WAP WML, Wireless Application Protocol, Wireless Markup Language Specification.* Version 1.2. WAP Forum, November 4, 1999
[WAPWML20]	WAP Forum: *Wireless Application Protocol, Wireless Markup Language, Version 2.0* WAP-238-WML-20010911-a. WAP Forum, September 11, 2001
[WELLNER]	Wellner, P.; Mackay, W.; Gold, R.: *Computer Augmented Environments: Back to the Real World*. Communications of the ACM 36 No. 7, 24-26 (July 1993).
[WELLNER2]	Wellner, P.: *Interacting with Paper on the Digital Desk*. Communications of the ACM 36 No. 7, 87-96 (July 1993).
[WERNEMAN]	Werneman, L.: *LS regarding User Profile*, 3GPP TSG-SA WG 1 (Services) meeting #12, May 2002.
[WSIREF]	Arbanowski, St.; Lipka, M.; Mössner, K.; Ott K.; Pabst R.; Pulli P.; Schieder, A.; Uusitalo, M. A.: *The WSI Reference Model for the Wireless World*. Proc. of the 21st IST summit on mobile and wireless communications, Aveiro, Portugal, June 15-18, 2003, Volume 2, pp. 613-617, ISBN 972-98368-7
[WU]	Wu, J.C.S.; His, E.C.N.; Kate, W.; Chen, P.M.C.: *A Framework for Web Content Adaptation*. W3C/WAP Workshop on Mulimodal Web, Philips Research, 2000
[X.901]	ITU-T Recommendation X.901: *Information technology - Open Distributed Processing - Reference Model: Overview*. - Geneva, Aug. 1997, (ISO/IEC IS 10746-1)
[X.902]	ITU-T Recommendation X.902: *Information technology - Open Distributed Processing - Reference Model: Foundations*. - Geneva, Nov. 1995, (ISO/IEC IS 10746-2)
[X.903]	ITU-T Recommendation X.903: *Information technology - Open Distributed Processing - Reference Model: Architecture*. - Geneva, Nov. 1995, (ISO/IEC IS 10746-3)
[X.904]	ITU-T Recommendation X.904: *Information technology - Open Distributed Processing - Reference Model: Architectural Semantics*. - Geneva, 1998, (ISO/IEC IS 10746-4)
[ZUE2]	Zue, V.; Seneff, S.; Glass J.; Hetherington L.; Hurley E.; Meng H.; Pao C.; Polifroni J.; Schloming R.; Schmid P.: *From Interface to Content: Translingual Access and Delivery of On-Line Information*. Spoken Language Systems Group, Laboratory for Computer Science, MIT, Cambridge, Massachusetts, USA

6.2 WWW-References

The Internet and the World Wide Web (WWW) have been a useful resource for creating this thesis. The following URLs have been valid until 11/25/2003. It is possible that they no longer exist at the time the reader studies this work.

[WWWAPACHE]	The Apache Software Foundation: www.apache.org
[WWWAOS]	FIPA Ontology Service Specification:. http://www.fipa.org/specs/fipa00086/
[WWWCOCOON]	The Apache XML Project: Apache Cocoon. Apache Software Foundation, 1999-2002. http://xml.apache.org/cocoon/index.html
[WWWBOV]	WWRF: Book of Visions, Edition December 2001; www.wireless-world-research.org/
[WWWDELI]	An open source Delivery Context Java library supporting CC/PP and UAProf (with HTTP header negotiation). http://sourceforge.net/projects/delicon/
[WWWGPS]	GPS Tutorial. http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html
[WWWIETFIPSEC]	Internet Engineering Task Force (IETF) IP security (IPsec) protocol Charter. http://www.ietf.org/html.charters/ipsec-charter.html
[WWWJBOSS]	JBoss Project Homepage. http://www.jboss.org
[WWWMOZILLA]	Mozilla Foundation. http://www.mozilla.org/
[WWWMSIE]	Microsoft Internet Explorer. http://www.microsoft.com/iexplorer
[WWWMYSQL]	MySQL AB: http://www.mysql.com/
[WWWOMA]	Open Mobile Alliance. http://www.oma.org
[WWWOSA]	OSA/Parlay Group. http://www.parlay.org/
[WWWP920]	Eurescom Project P920: *UMTS Networks Aspects*. http://www.eurescom.de/public/projects/P900-series/p920/
[WWWPIAVIDA]	PI-AVIda - Personalisierte Interaktive Audio-, Voice- und Video- Informationen für Portal und Auswertungsanwendungen. 2003. http://www.igd.fhg.de/igd-a7/piavida/
[WWWQUICKT]	Apple. Quicktime Player. http://www.apple.com
[WWWREALPL]	Real Networks. The RealOne Player. http://www.real.com
[WWWRSAPKCS]	RSA Public-Key Cryptography Standards (PKCS) Overview. http://www.rsasecurity.com/rsalabs/pkcs
[WWWSAX]	Simple API for XML (SAX) Project Homepage. http://www.saxproject.org/
[WWWSEMW]	Semantic Web Community Portal. http://www.semanticweb.org/
[WWWSMARTDRAW]	How to Draw UML Diagrams. Smartdraw.com UML Center, 2002. http://www.smartdraw.com/resources/centers/uml/uml.htm
[WWWTINI]	TINI: Tiny InterNet Interface. http://www.iButton.com/TINI/
[WWWTOMCAT]	The Apache Jakarta Project: Apache Tomcat, Apache Software Foundation, 1999-2002. http://jakarta.apache.org/tomcat/index.html
[WWWUMTSFORUM]	UTMS Forum Homepage. http://www.umts-forum.org
[WWWVESPER]	IST -10985 VESPER project. http://vesper.intranet.gr
[WWWVHEUD]	Fraunhofer FOKUS, ISST: VHE und Dienste. 2003. http://www.isst.fhg.de/german/projekte/2002/vhe.html
[WWWW3C]	World Wide Web Consortium. http://www.w3c.org
[WWWWG2BM]	WWRF Working Group 2: White Paper on Business Models
[WWWWG2PER]	WWRF Working Group 2: White Paper on Service Personalization
[WWWWG2AA]	WWRF Working Group 2: White Paper on Ambient Awareness
[WWWWG2AD]	WWRF Working Group 2: White Paper on Service Adaptability
[WWWWGS84]	World Geodetic System 1984, http://www.wgs84.com/
[WWWWRF]	Wireless World Research Forum. http://www.wireless-world-research.org/
[WWWWRI]	IST Wireless Strategic Initiative. http://www.ist-wsi.org/
[WWWWSI]	IST Wireless Strategic Initiative. http://www.ist-wsi.org/
[WWWXHTML]	Miloslav Nic: XHTML Basic reference with examples. Systinet, 2000. http://www.zvon.org/xxl/xhtmlBasicReference/Output/

6.3 Acronyms

3G	Third Generation
3Gb	Third Generation beyond
3GPP	Third Generation Partnership Project
AAA	Authentication, Authorization, Accounting
ACM	Association for Computing Machinery
AIS	Ambient Information System
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
ASR	Automatic Speech Recognition
AU	Audio-Basic-File
B2C	Business-to-Customer
BAN	Body Area Network
BMBF	Bundesministerium für Bildung und Forschung
BoV	Books of Vision
CC/PP	Composite Capabilities / Preferences Profile
cHTML	compact Hypertext Markup Language
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
CSS	Cascading Style Sheet
DDF	Data Definition Framework
DDL	Dialog Description Language
DHTML	Dynamic Hypertext Markup Language
DI	Device Independence
DNS	Domain Name System
DOM	Document Object Model
DPE	Distributed Processing Environment
DSL	Digital Subscribe Line
DTD	Document Type Definition
DTMF	Dual Tone Multi Frequency
EJB	Enterprise Java Beans
e-mail	electronic Mail
EMS	Enhanced Messaging Service (Extension of SMS)
ETSI	European Telecommunications Standards Institute
FOKUS	Forschungsinstitut für Offene Kommunikationssysteme
GIF	Graphics Interchange Format
GPRS	General Packet Radio Service
GPS	Global Positioning System
GPS	General Preference Set
GSM	Global System for Mobile Communication
GSM-MoU	Global System for Mobile Communication Memorandum of Understanding
GUI	Graphical User Interface
GUIML	Generic User Interaction Markup Language
GUP	Generic User Profile
HE	Home Environment
HE-VASP	Home Environment Value-Added Service Provider
HMI	Human Machine Interface
HSCSD	High Speed Circuit Switched Data
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol with Secure Socket Layer
ID	Identifier
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IM	Instant Messaging
IMT 2000	International Mobile Telecommunication 2000
IP	Internet Protocol
IPsec	Internet Protocol Security Extension
IrDA	Infrared Device Access
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
IST	Information Society Technologies
ISTAG	Information Society Technologies Advisory Group
ITU	International Telecommunication Union
IVR	Interactive Voice Response
J2SE	Java 2 Standard Edition
JDBC	Java Data Base Connectivity
JDK	Java Development Kit
JMF	Java Media Framework
JNDI	Java Naming and Directory Service
JPEG	Joint Photographic Experts Group
JSP	Java Server Pages
LAN	Local Area Network
MDA	Model Driven Architecture
MExE	Mobile Execution Environment
MMS	Multimedia Messaging Service
MPEG	Motion Picture Expert Group
MSC	Message Sequence Chart
NITF	News Industry Text Format
ODBC	Open Database Access
ODP	Open Distributed Processing
OMA	Open Mobile Alliance
OMG	Object Management Group
OSA	Open Service Access
OSI	Open System Interconnection
P2P	Peer-to-peer
P3P	Platform for Privacy Preferences Project
PAN	Personal Area Network
PC	Personal Computer
PDA	Personal Digital Assistant
PIM	Platform Independent Model
PKCS	Public Key Cryptography Standard
PMS	Preferences Management System
PSE	Personal Service Environment
PSM	Platform Specific Model

PSTN	Public Switched Telephony Network
QoS	Quality of Service
RDF	Resource Description Framework
RFC	Request for Comments
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RSA	Rivest-Shamir-Adleman
SAF	Service Adaptation Function
SAX	Simple API for XML
SDO	Super Distributed Object
SLA	Service Level Agreement
SMIL	Synchronized Multimedia Integration Language
SMS	Short Messaging Service
SOAP	Simple Object Access Protocol
SP	Service Preference
SPS	Service Preference Set
SQL	Structured Query Language
SSL	Secure Socket Layer
SVG	Scalable Vector Graphics
TINA	Telecommunication Information Networking Architecture
TS	Technical Specification
TTS	Text-To-Speech
TUB	Technical University Berlin
TV	Television
UAProf	User Agent Profile
UI	User Interface
UIML	User Interface Markup Language
UML	Unified Modeling Language
UMS	Unified Messaging System
UMTS	Universal Mobile Telecommunications System
UP	User Profile
UR	User Record
URI	Universal Resource Identifier
URL	Universal/Uniform Resource Location
UUID	Universal Unique IDentifer
VASP	Value-Added Service Provider
VHE	Virtual Home Environment
VHE-UD	VHE-Umgebung und Dienste
VoIP	Voice over Internet Protocol
VXML	Voice Extensible Markup Language
W3C	World Wide Web Consortium
WAG	Wireless Application Group
WAN	Wide Area Network
WAP	Wireless Application Protocol
WAV	Wave Audio Format
WBMP	Wireless Bit Map
wLAN	wireless Local Area Network
WML	Wireless Markup Language
WSDD	Web Service Deployment Descriptor
WSDL	Web Service Description Language
WWRF	Wireless World Research Forum
WWRI	Wireless World Research Initiative
WWW	World Wide Web
XCDL	Extensible Delivery Context Language
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations

CHAPTER 7 Appendix

This chapter contains some additional information usable to understand the work presented in this thesis as well as the complete specifications, which cannot be presented in the particular sections because of their size.

7.1 Complete GUIML Specification

This section contains the Document Type Definition (DTD) for the Generic User Interaction Markup Language (GUIML). The complete specification consists of the following five documents and refers to further modules, provided by www.w3.org:

- guiml-1_0.dtd
- guiml-model-1.mod
- additional-guiml-elements-1.mod
- xforms-basic-qname-1.mod
- xforms-basic-elements-1.mod

The five documents are supposed to reside at the same location, e.g. directory. Unfortunately, DTDs cannot process the tags of the XForms instance variables. If XForms instance variables are included in a GUIML document, the validation against this DTD has to be turned off. This is because of a technical limitation of DTDs, which cannot ignore and skip unknown tags. That is why at time of writing this thesis, this specification was being translated into an according XML Schema specification. The exact XHTML and SMIL element definitions can be found in the referenced W3C modules.

In this thesis, the initial DTD-based specification is presented, because the transformation to the XML Schema specification was not yet finished and successfully tested.

7.1.1 'guiml-1_0.dtd'

This file is the DTD driver for GUIML. It has to be included into the GUIML document after the standard XML header referencing this file:

```
<!DOCTYPE document SYSTEM "guiml-1_0.dtd">
```

It calls all modules and defines the SMIL test attributes. The definition has to be done here because there is a conflict with the 'Core.attribs' entities in SMIL and XHTML.

```
<!-- GUIML DTD  ............................................................ -->
<!-- file: guiml-1_0.dtd -->
<!--
This is the DTD driver for GUIML 1.0.
Please use this formal public identifier to identify it:
  "-//FOKUS//DTD XHTML GUIML 1.0//EN"
-->
<!ENTITY % XHTML.version "-//FOKUS//DTD XHTML GUIML 1.0//EN">
<!-- reserved for use with document profiles -->
<!ENTITY % XHTML.profile "">
<!-- Tell the framework to use our qualified names module as an extra qname driver -->
```

```
<!--<!ENTITY % xhtml-qname-extra.mod SYSTEM "guiml-qname-1.mod">-->
<!-- Define the Content Model for the framework to use -->
<!--<!ENTITY % xhtml-model.mod SYSTEM "guiml-model-1.mod">-->
<!-- Disable bidirectional text support -->
<!ENTITY % XHTML.bidi "IGNORE">
<!-- XForms Elements  -->
<!ENTITY % XForms-basic.prefixed "INCLUDE">
<!ENTITY % XForms-basic.prefix "xfm">
<!ENTITY % xhtml-qname-extra.mod SYSTEM "xforms-basic-qname-1.mod">
<!ENTITY % xforms-basic-elements.mod SYSTEM "xforms-basic-elements-1.mod">
%xforms-basic-elements.mod;
<!-- SMIL -->
<!-- SMIL Framework -->
<!ENTITY % SMIL.prefixed "INCLUDE">
<!ENTITY % SMIL.prefix "smil">
<!ENTITY % smil-model.module "IGNORE">
<!ENTITY % smil-attribs.module "IGNORE">
<!ENTITY % smil-framework.mod PUBLIC "-//W3C//ENTITIES SMIL 2.0 Modular Framework 1.0//EN"
  "http://www.w3.org/2001/SMIL20/smil-framework-1.mod">
%smil-framework.mod;
<!ENTITY % smil-model.mod PUBLIC "-//W3C//ENTITIES SMIL 2.0 Document Model 1.0//EN"
 "http://www.w3.org/2001/SMIL20/smil-model-1.mod">
%smil-model.mod;
<!-- Direct insertion of Test Attributes because of Core.attribs conflict of SMIL n XHTML -->
<!ENTITY % SMIL.customTestAttr.attrib "
 %SMIL.pfx;customTest IDREF #IMPLIED">
<!ENTITY % SMIL.skip-content.attrib "
 %SMIL.pfx;skip-content (true|false) 'true'">
<!ENTITY % SMIL.Test.attrib "
 %SMIL.pfx;systemBitrate CDATA #IMPLIED
 %SMIL.pfx;systemCaptions (on|off) #IMPLIED
 %SMIL.pfx;systemLanguage CDATA #IMPLIED
 %SMIL.pfx;systemOverdubOrSubtitle (overdub|subtitle) #IMPLIED
 %SMIL.pfx;systemRequired CDATA #IMPLIED
 %SMIL.pfx;systemScreenSize CDATA #IMPLIED
 %SMIL.pfx;systemScreenDepth CDATA #IMPLIED
 %SMIL.pfx;systemAudioDesc (on|off) #IMPLIED
 %SMIL.pfx;systemOperatingSystem NMTOKEN #IMPLIED
 %SMIL.pfx;systemCPU NMTOKEN #IMPLIED
 %SMIL.pfx;systemComponent CDATA #IMPLIED
 %SMIL.pfx;system-bitrate CDATA #IMPLIED
 %SMIL.pfx;system-captions (on|off)  #IMPLIED
 %SMIL.pfx;system-language CDATA #IMPLIED
 %SMIL.pfx;system-overdub-or-caption (overdub|caption) #IMPLIED
 %SMIL.pfx;system-required CDATA #IMPLIED
 %SMIL.pfx;system-screen-size CDATA #IMPLIED
 %SMIL.pfx;system-screen-depth CDATA #IMPLIED
">
<!ENTITY % style.attrib "style CDATA #IMPLIED">
<!ENTITY % Core.extra.attrib "
 %style.attrib;
 %SMIL.customTestAttr.attrib;
 %SMIL.skip-content.attrib;
 %SMIL.Test.attrib;">
<!-- SMIL modules -->
<!ENTITY % SMIL.control-mod PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Content Control//EN"
  "http://www.w3.org/2001/SMIL20/SMIL-control.mod">
%SMIL.control-mod;
<!ENTITY % SMIL.media-mod PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Media Objects//EN"
  "http://www.w3.org/2001/SMIL20/SMIL-media.mod">
%SMIL.media-mod;
<!-- GUIML Model (SMIL, Xforms, Frames) DRIVER FILE, call before XHTML Framework-->
<!ENTITY % xhtml-model.mod PUBLIC "-//W3C//ENTITIES XHTML Basic 1.0 Document Model 1.0//EN"
 "guiml-model-1.mod">
<!-- Bring in the XHTML Framework -->
<!ENTITY % xhtml-framework.mod PUBLIC "-//W3C//ENTITIES XHTML Modular Framework 1.0//EN"
 "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-framework-1.mod">
%xhtml-framework.mod;
<!-- Insert SMIL attribs after XHTML Framework because of core.attribs conflicts -->
<!ENTITY % smil-attribs.mod PUBLIC "-//W3C//ENTITIES SMIL 2.0 Common Attributes 1.0//EN"
 "http://www.w3.org/2001/SMIL20/smil-attribs-1.mod">
%smil-attribs.mod;
<!-- Basic Text Module (Required)  ............................... -->
<!ENTITY % xhtml-text.mod PUBLIC "-//W3C//ELEMENTS XHTML Basic Text 1.0//EN"
 "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-text-1.mod">
%xhtml-text.mod;
<!-- Hypertext Module (required) ................................ -->
<!ENTITY % xhtml-hypertext.mod PUBLIC "-//W3C//ELEMENTS XHTML Hypertext 1.0//EN"
 "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-hypertext-1.mod">
%xhtml-hypertext.mod;
<!-- Lists Module (required)  ................................... -->
<!ENTITY % xhtml-list.mod PUBLIC "-//W3C//ELEMENTS XHTML Lists 1.0//EN"
  "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-list-1.mod">
%xhtml-list.mod;
<!-- Basic Tables Module ............................................. -->
<!ENTITY % xhtml-table.mod PUBLIC "-//W3C//ELEMENTS XHTML Basic Tables 1.0//EN"
 "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-basic-table-1.mod">
%xhtml-table.mod;
<!-- Link Element Module  ....................................... -->
<!ENTITY % xhtml-link.mod PUBLIC "-//W3C//ELEMENTS XHTML Link Element 1.0//EN"
  "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-link-1.mod">
%xhtml-link.mod;
<!-- Base Element Module  ....................................... -->
<!ENTITY % xhtml-base.mod PUBLIC "-//W3C//ELEMENTS XHTML Base Element 1.0//EN"
 "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-base-1.mod">
%xhtml-base.mod;
<!-- Document Structure Module (required)  ...................... -->
<!ENTITY % xhtml-struct.mod PUBLIC "-//W3C//ELEMENTS XHTML Document Structure 1.0//EN"
 "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-struct-1.mod">
%xhtml-struct.mod;
```

```
<!-- Adiitional GUIML Elements  -->
<!ENTITY % additional-guiml-elements.mod SYSTEM "additional-guiml-elements-1.mod">
%additional-guiml-elements.mod;
<!-- end of guiml-1_0.dtd -->
```

7.1.2 ‘guiml-model-1.mod’

This is the actual GUIML core model.

```
<!-- GUIML Model Module  .................................................. -->
<!-- file: guiml-model-1.mod
     PUBLIC "-//MY COMPANY//ELEMENTS XHTML GUIML Model 1.0//EN"
     SYSTEM "http://example.com/DTDs/guiml-model-1_0.mod"
     xmlns:guiml="http://www.example.com/xmlns/guiml"
     ................................................................... -->
<!-- Define the content model for Misc.extra -->
<!ENTITY % html.element "IGNORE">
<!ENTITY % Misc.class "| frame  | %SMIL.switch.qname; | %SMIL.ref.qname; | %SMIL.audio.qname; |
 %SMIL.img.qname; | %SMIL.video.qname; | %SMIL.text.qname; | %SMIL.textstream.qname; |
 %SMIL.animation.qname; | %XForms-basic.model.qname; | %XForms-basic.input.qname; | %XForms-
 basic.secret.qname; | %XForms-basic.textarea.qname; | %XForms-basic.trigger.qname; | %XForms-
 basic.submit.qname; | %XForms-basic.select1.qname; | %XForms-basic.select.qname;">

<!-- Optional Elements in head  ............. -->
<!ENTITY % HeadOpts.mix "( %meta.qname; | %link.qname; | %object.qname; )*">
<!-- Miscellaneous Elements  ................ -->
<!ENTITY % Misc.class "">
<!-- Inline Elements  ....................... -->
<!ENTITY % InlStruct.class "%br.qname; | %span.qname;">
<!ENTITY % InlPhras.class "| %em.qname; | %strong.qname; | %dfn.qname; | %code.qname;
      | %samp.qname; | %kbd.qname; | %var.qname; | %cite.qname;
      | %abbr.qname; | %acronym.qname; | %q.qname;">
<!ENTITY % InlPres.class "">
<!ENTITY % I18n.class "">
<!ENTITY % Anchor.class "| %a.qname;">
<!ENTITY % InlSpecial.class "| %img.qname; | %object.qname;">
<!ENTITY % InlForm.class "| %input.qname; | %select.qname; | %textarea.qname;
      | %label.qname;">
<!ENTITY % Inline.extra "">
<!ENTITY % Inline.class "%InlStruct.class;
      %InlPhras.class;
      %Anchor.class;
      %InlSpecial.class;
      %InlForm.class;
      %Inline.extra;">
<!ENTITY % InlNoAnchor.class "%InlStruct.class;
      %InlPhras.class;
      %InlSpecial.class;
      %InlForm.class;
      %Inline.extra;">
<!ENTITY % InlNoAnchor.mix "%InlNoAnchor.class;
      %Misc.class;">
<!ENTITY % Inline.mix "%Inline.class;
      %Misc.class;">
<!-- Block Elements  ........................ -->
<!ENTITY % Heading.class "%h1.qname; | %h2.qname; | %h3.qname;
      | %h4.qname; | %h5.qname; | %h6.qname;">
<!ENTITY % List.class "%ul.qname; | %ol.qname; | %dl.qname;">
<!ENTITY % Table.class "| %table.qname;">
<!ENTITY % Form.class "| %form.qname;">
<!ENTITY % BlkStruct.class "%p.qname; | %div.qname;">
<!ENTITY % BlkPhras.class "| %pre.qname; | %blockquote.qname; | %address.qname;">
<!ENTITY % BlkPres.class "">
<!ENTITY % BlkSpecial.class "%Table.class;
      %Form.class;">
<!ENTITY % Block.extra "">
<!ENTITY % Block.class "%BlkStruct.class;
      %BlkPhras.class;
      %BlkSpecial.class;
      %Block.extra;">
<!ENTITY % Block.mix "%Heading.class;
      | %List.class;
      | %Block.class;
      %Misc.class;">
<!-- All Content Elements  .................. -->
<!-- declares all content except tables -->
<!ENTITY % FlowNoTable.mix "%Heading.class;
      | %List.class;
      | %BlkStruct.class;
      %BlkPhras.class;
      %Form.class;
      %Block.extra;
      | %Inline.class;
      %Misc.class;">
<!ENTITY % Flow.mix "%Heading.class;
      | %List.class;
      | %Block.class;
      | %Inline.class;
      %Misc.class;">
<!-- end of guiml-model-1.mod -->
```

7.1.3 ‘additional-guiml-elements-1.mod’

This module defines the GUIML elements ‘document’ and ‘frame’.

```
<!-- Additional GUIML Elements Module ................................................ -->
<!-- file: additional-guiml-elements-1.mod
     PUBLIC "-//FOKUS//ELEMENTS XHTML Additional GUIML Elements 1.0//EN"
     SYSTEM "http://www.fokus.fraunhofer.de/DTDs/additional-guiml-elements-1_0.mod"
     xmlns:xfm="http://www.fokus.fraunhofer.de/additional-guiml"-->
<!-- Document (Root) -->
<!ENTITY % document.content "( %head.qname;, %body.qname; )">
<!ENTITY % document.qname "document">
<!ELEMENT %document.qname; %document.content;>
<!-- Frame -->
<!ENTITY % frame.content "ANY">
<!ENTITY % frame.qname "frame">
<!ELEMENT %frame.qname; %frame.content;>
<!ATTLIST %frame.qname;
 id CDATA #IMPLIED
 title CDATA #IMPLIED
 style CDATA #IMPLIED
 href CDATA #IMPLIED
 xml:base CDATA #IMPLIED
>
<!-- end of additional-guiml-elements-1.mod -->
```

7.1.4 ‘xforms-basic-qname-1.mod’

This module defines the qualified name for XForms Basic.

```
<!-- file: xforms-basic-qname-1.mod -->
<!-- Bring in the datatypes - we use the URI.datatype PE for declaring the
     xmlns attributes. -->
<!ENTITY % XForms-basic-datatypes.mod PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
          "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-datatypes-1.mod">
%XForms-basic-datatypes.mod;
<!-- By default, disable prefixing of this module -->
<!ENTITY % NS.prefixed "IGNORE">
<!ENTITY % XForms-basic.prefixed "%NS.prefixed;">
<!-- Declare the actual namespace of this module -->
<!ENTITY % XForms-basic.xmlns "http://www.fokus.fraunhofer.de/xforms-basic">
<!-- Declare the default prefix for this module -->
<!ENTITY % XForms-basic.prefix "xfm">
<!-- If this module's namespace is prefixed -->
<!%XForms-basic.prefixed;
  <!ENTITY % XForms-basic.pfx  "%XForms-basic.prefix;:" >
]]>
<!ENTITY % XForms-basic.pfx "">
<!-- Declare a Parameter Entity (PE) that defines any external namespaces
     that are used by this module -->
<!ENTITY % XForms-basic.xmlns.extra.attrib "">
<!-- Declare a PE that defines the xmlns attributes for use by Forms Basic. -->
<!%XForms-basic.prefixed;
<!ENTITY % XForms-basic.xmlns.attrib
   "xmlns:%XForms-basic.prefix;  %URI.datatype;  #FIXED '%XForms-basic.xmlns;'
   %XForms-basic.xmlns.extra.attrib;"
>
]]>
<!ENTITY % XForms-basic.xmlns.attrib "xmlns   %URI.datatype;  #FIXED '%XForms-basic.xmlns;'
    %XForms-basic.xmlns.extra.attrib;">
<!-- Make sure that the XForms basic namespace attributes are included on the XHTML
     attribute set -->
<!%NS.prefixed;
<!ENTITY % XHTML.xmlns.extra.attrib
    "%XForms-basic.xmlns.attrib;" >
]]>
<!ENTITY % XHTML.xmlns.extra.attrib "">
<!-- Now declare the element names -->
<!ENTITY % XForms-basic.label.qname "%XForms-basic.pfx;label">
<!ENTITY % XForms-basic.load.qname "%XForms-basic.pfx;load">
<!ENTITY % XForms-basic.action.qname "%XForms-basic.pfx;action">
<!ENTITY % XForms-basic.item.qname "%XForms-basic.pfx;item">
<!ENTITY % XForms-basic.value.qname "%XForms-basic.pfx;value">
<!ENTITY % XForms-basic.send.qname "%XForms-basic.pfx;send">
<!ENTITY % XForms-basic.help.qname "%XForms-basic.pfx;help">
<!ENTITY % XForms-basic.hint.qname "%XForms-basic.pfx;hint">
<!ENTITY % XForms-basic.model.qname "%XForms-basic.pfx;model">
<!ENTITY % XForms-basic.submission.qname "%XForms-basic.pfx;submission">
<!ENTITY % XForms-basic.instance.qname "%XForms-basic.pfx;instance">
<!ENTITY % XForms-basic.input.qname "%XForms-basic.pfx;input">
<!ENTITY % XForms-basic.secret.qname "%XForms-basic.pfx;secret">
<!ENTITY % XForms-basic.textarea.qname "%XForms-basic.pfx;textarea">
<!ENTITY % XForms-basic.trigger.qname "%XForms-basic.pfx;trigger">
<!ENTITY % XForms-basic.submit.qname "%XForms-basic.pfx;submit">
<!ENTITY % XForms-basic.select1.qname "%XForms-basic.pfx;select1">
<!ENTITY % XForms-basic.select.qname "%XForms-basic.pfx;select">
<!-- end of xforms-basic-qname-1.mod -->
```

7.1.5 ‘xforms-basic-elements-1.mod’

This module defines the XForms Basic elements. This is a self-defined subset of XForms because the XForms Basic standardization was not yet finished at time of writing.

```
<!-- XForms Basic Elements Module ................................................ -->
<!-- file: xforms-basic-elements-1.mod
     PUBLIC "-//FOKUS//ELEMENTS XHTML XForms Basic Elements 1.0//EN"
     SYSTEM "http://www.fokus.fraunhofer.de/DTDs/xforms-basic-elements-1_0.mod"
     xmlns:xfm="http://www.fokus.fraunhofer.de/xforms-basic"-->
```

```

<!-- Label -->
<!ELEMENT %XForms-basic.label.qname; ANY>
<!ATTLIST %XForms-basic.label.qname;
 %XForms-basic.xmlns.attrib;
>
<!-- Help -->
<!ELEMENT %XForms-basic.help.qname; (#PCDATA)>
<!ATTLIST %XForms-basic.help.qname;
 %XForms-basic.xmlns.attrib;
>
<!-- Hint -->
<!ELEMENT %XForms-basic.hint.qname; (#PCDATA)>
<!ATTLIST %XForms-basic.hint.qname;
 %XForms-basic.xmlns.attrib;
>
<!-- Value -->
<!ELEMENT %XForms-basic.value.qname; (#PCDATA)>
<!ATTLIST %XForms-basic.value.qname;
 %XForms-basic.xmlns.attrib;
>
<!-- Load -->
<!ELEMENT %XForms-basic.load.qname; EMPTY>
<!ATTLIST %XForms-basic.load.qname;
 %XForms-basic.xmlns.attrib;
 xlink:href CDATA #REQUIRED
>
<!-- Send -->
<!ELEMENT %XForms-basic.send.qname; EMPTY>
<!ATTLIST %XForms-basic.send.qname;
 %XForms-basic.xmlns.attrib;
>
<!-- Item -->
<!ELEMENT %XForms-basic.item.qname; ((%XForms-basic.label.qname;), (%XForms-basic.help.qname;)?,
 (%XForms-basic.hint.qname;)?, (%XForms-basic.action.qname; | %XForms-basic.value.qname;))>
<!ATTLIST %XForms-basic.item.qname;
 %XForms-basic.xmlns.attrib;
 %Core.attrib;
>
<!-- Action -->
<!ELEMENT %XForms-basic.action.qname; ((%XForms-basic.load.qname;) | (%XForms-
basic.send.qname;))>
<!ATTLIST %XForms-basic.action.qname;
 %XForms-basic.xmlns.attrib;
>
<!-- Model -->
<!ELEMENT %XForms-basic.model.qname; ((%XForms-basic.submission.qname;), (%XForms-
basic.instance.qname;)?)>
<!ATTLIST %XForms-basic.model.qname;
 %XForms-basic.xmlns.attrib;
 id CDATA #REQUIRED
>
<!-- Submission -->
<!ELEMENT %XForms-basic.submission.qname; EMPTY>
<!ATTLIST %XForms-basic.submission.qname;
 %XForms-basic.xmlns.attrib;
 action CDATA #REQUIRED
 id CDATA #IMPLIED
 method (get | post) #REQUIRED
>
<!-- Instance -->
<!ELEMENT %XForms-basic.instance.qname; ANY>
<!ATTLIST %XForms-basic.instance.qname;
 %XForms-basic.xmlns.attrib;
>
<!-- Controls -->
<!-- Common attributes -->
<!ENTITY % XForms-basic.common.attrib "
 model  CDATA #REQUIRED
 ref CDATA #REQUIRED
">
<!-- Input -->
<!ELEMENT %XForms-basic.input.qname; ((%XForms-basic.label.qname;), (%XForms-basic.help.qname;)?,
 (%XForms-basic.hint.qname;)?)>
<!ATTLIST %XForms-basic.input.qname;
 %XForms-basic.xmlns.attrib;
 %XForms-basic.common.attrib;
>
<!-- Secret -->
<!ELEMENT %XForms-basic.secret.qname; ((%XForms-basic.label.qname;), (%XForms-
basic.help.qname;)?, (%XForms-basic.hint.qname;)?)>
<!ATTLIST %XForms-basic.secret.qname;
 %XForms-basic.xmlns.attrib;
 %XForms-basic.common.attrib;
>
<!-- Textarea -->
<!ELEMENT %XForms-basic.textarea.qname; ((%XForms-basic.label.qname;), (%XForms-
basic.help.qname;)?, (%XForms-basic.hint.qname;)?)>
<!ATTLIST %XForms-basic.textarea.qname;
 %XForms-basic.xmlns.attrib;
 %XForms-basic.common.attrib;
>
<!-- Select -->
<!ELEMENT %XForms-basic.select.qname; ((%XForms-basic.label.qname;), (%XForms-
basic.help.qname;)?, (%XForms-basic.hint.qname;)?, (%XForms-basic.item.qname;)+)>
<!ATTLIST %XForms-basic.select.qname;
 %XForms-basic.xmlns.attrib;
 %XForms-basic.common.attrib;
>
<!-- Select1 -->
```

```
<!ELEMENT %XForms-basic.select1.qname; ((%XForms-basic.label.qname;), (%XForms-
    basic.help.qname;)?, (%XForms-basic.hint.qname;)?, (%XForms-basic.item.qname;)+)>
<!ATTLIST %XForms-basic.select1.qname;
 %XForms-basic.xmlns.attrib;
 %XForms-basic.common.attrib;
>
<!-- Trigger -->
<!ELEMENT %XForms-basic.trigger.qname; ((%XForms-basic.label.qname;), (%XForms-
    basic.help.qname;)?, (%XForms-basic.hint.qname;)?, (%XForms-basic.action.qname;))>
<!ATTLIST %XForms-basic.trigger.qname;
 %XForms-basic.xmlns.attrib;
 model CDATA #REQUIRED
>
<!-- Submit -->
<!ELEMENT %XForms-basic.submit.qname; (%XForms-basic.label.qname;, (%XForms-basic.help.qname;)?,
    (%XForms-basic.hint.qname;)?)>
<!ATTLIST %XForms-basic.submit.qname;
 %XForms-basic.xmlns.attrib;
  model  CDATA #REQUIRED
  ref CDATA #IMPLIED
>
<!-- end of xforms-basic-elements-1.mod -->
```

7.2 Dialog Adaptation Templates

The following sections contain the dialog adaptation templates for HTML, WML, and VoiceXML. These templates are used in the implementation of the Dialog Adaptor component of the I-centric User Interaction Portal (section 4.6.5).

7.2.1 Dialog Adaptation Templates for HTML

Template	*Description*
match=”frame|body”	inserts a form for each model and the predefined values as hidden parameters
match=”@*|node()”	calls the template ‘xformcontrols’ if an element has the prefix ‘xfm’
name=”xformcontrols”	calls the named templates for all XForms controls
name=”submit”	adapts the XForms submit control
name=”input”	adapts the XForms input control
name=”secret”	adapts the XForms secret control
name=”textarea”	adapts the XForms textarea control
name=”trigger”	adapts the XForms trigger control
name=”select1”	adapts the XForms select1 control
name=”select”	adapt the XForms select control

7.2.2 Dialog Adaptation Templates for WML

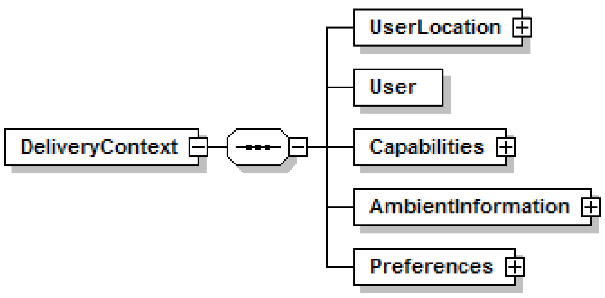
Template	*Description*
match=”frame|body”	inserts the predefined values as predefined variables
match=”@*|node()”	calls the template xformcontrols if an element has the prefix xfm
name=”xformcontrols”	calls the named templates for all XForms controls
name=”insertVariables”	inserts all variables in the format of GET request parameters
name=”submit”	adapts the XForms submit control
name=”input”	adapts the XForms input control
name=”secret”	adapts the XForms secret control
name=”textarea”	adapts the XForms textarea control
name=”trigger”	adapts the XForms trigger control
name=”select1”	adapts the XForms select1 control
name=”select”	adapts the XForms select control

7.2.3 Dialog Adaptation Templates for VoiceXML

Template	***Description***
match=”body”	distinguishes between a frameless and a document using frames and calls the following templates: ‘insertGlobalLinks’, ‘insertExplanationLinks’, and ‘insertEndOfPageForm’
match=”frame” name=”frame”	inserts a form for each XForms model and adds predefined values as predefined variables
match=”@*|node()”	calls the template xforms if an element has the prefix xfm
name=”xforms”	calls the named templates for all XForms controls
name=”submit”	adapts the XForms submit control. Therefore, the template ‘insertSubmit’ is called
name=”input”	adapts the XForms input control. Digits are allowed as input
name=”secret”	adapts the XForms secret control, uses the same template as the input control
name=”textarea”	adapts the XForms textarea control, uses the same template as the input control
name=”trigger”	adapts the XForms trigger control
name=”select1”	adapts the XForms select1 control. Therefore, a grammar with all options is built, from which the user can select one
name=”select”	adapt the XForms select control. Therefore, a grammar with all options is built, from which the user can select as many as he wants until he says “complete”
name=”insertSubmit”	inserts a VoiceXML submit including a namelist filled with all variables
name=”change_predefined”	inserts a question to the user that asks him if he wants to change a predefined value for a variable. This template is called at the beginning of some XForms control adaptation
name=”insertGlobalLinks”	inserts global links and grammars to all available targets, i.e. links, menus and frames (scopes). insertLinksToAllTargets is used as helper template
name=”insertExplanationLinks”	inserts all available target names for user help. Uses linkselector, menuselector and frameselector as helper templates
name=”insertEndOfPageForm”	inserts all available target names at the end of the VoiceXML dialog to request the selection of the next target from the user
name=”insertLinksToAllTargets”	inserts all available targets (derived from trigger and select1 controls containing a load action) in the current frame context
name=”linkselector”	calls insertlinknames for the current frame or body, i.e. the names of all links are put in
name=”menuselector”	calls insertlinknames for all available menu frames, i.e. the names of all links at menu frames are put in
name=”frameselector”	inserts the name of all frames except the current frame. The enables the user to get the list of all available frames
name=”insertlinknames”	browses all available controls with load actions in the current scope to insert their labels

7.3 Delivery Context Specification – ‘deliverycontext.xsd’

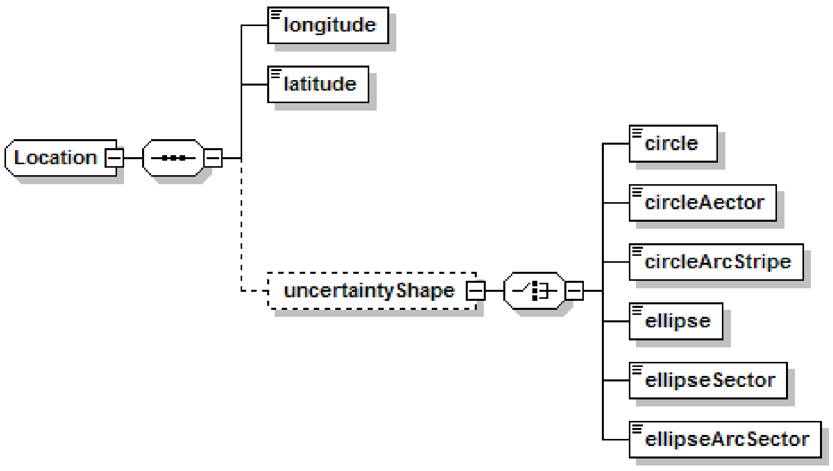
The following sections present the complete specification of the deliver context. As is can be seen in the following main part, the specification consists of ‘UserLocation’, ‘Capabilities’, ‘AmbientInformation’, and ‘Preferences’ module, which are presented in separated sections. The ‘User’ element depicts a generic user identifier, whose type to be defined by the corresponding implementation.



```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xdcl="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:include schemaLocation="geographicalposition.xsd"/>
 <xs:include schemaLocation="preferences.xsd"/>
 <xs:include schemaLocation="ambientinformation.xsd"/>
 <xs:import namespace="xdcl" schemaLocation="XDCL/xdcl-1.0.xsd"/>
 <xs:element name="DeliveryContext">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="UserLocation" type="Location"/>
            <xs:element name="User"/>
            <xs:element name="Capabilities" type="xdcl:Profile"/>
            <xs:element name="AmbientInformation" type="AmbientProfile"/>
            <xs:element name="Preferences" type="UserPreferences"/>
        </xs:sequence>
    </xs:complexType>
 </xs:element>
</xs:schema>
```

7.3.1 Location Module – 'geographicalposition.xsd'

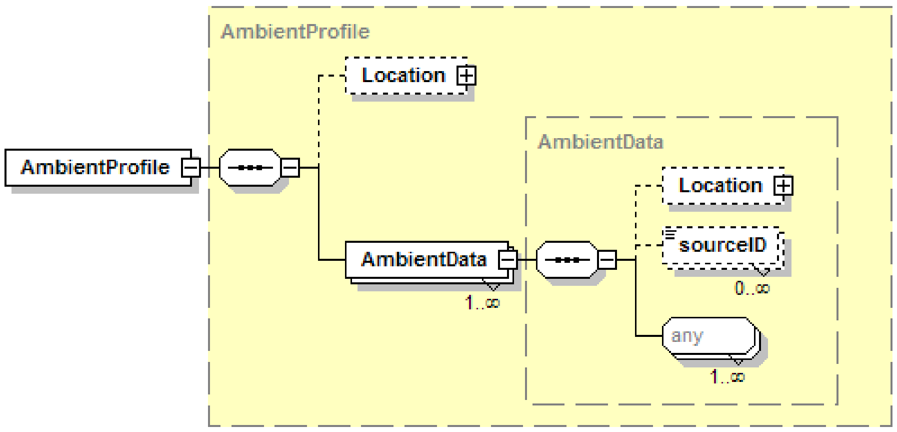
This module defines the specification of location references. The conceptual model was derived from [WWWWG84], which is also adopted in the Parlay specification [PRLY21MOB].



```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
 attributeFormDefault="unqualified">
 <xs:attribute name="unit" use="required">
    <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="meter"/>
            <xs:enumeration value="foot"/>
        </xs:restriction>
    </xs:simpleType>
 </xs:attribute>
 <xs:complexType name="Location">
    <xs:sequence>
        <xs:element name="longitude" type="xs:string"/>
        <xs:element name="latitude" type="xs:string"/>
        <xs:element name="uncertaintyShape" minOccurs="0">
            <xs:complexType>
                <xs:choice>
                    <xs:element name="circle">
                        <xs:complexType>
                            <xs:simpleContent>
                                <xs:extension base="xs:double">
                                    <xs:attribute name="unit" use="required">
                                        <xs:simpleType>
                                            <xs:restriction base="xs:NMTOKEN">
                                                <xs:enumeration value="meter"/>
                                                <xs:enumeration value="foot"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:attribute>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="circleAector" type="xs:long"/>
            <xs:element name="circleArcStripe" type="xs:long"/>
            <xs:element name="ellipse" type="xs:long"/>
            <xs:element name="ellipseSector" type="xs:long"/>
            <xs:element name="ellipseArcSector" type="xs:long"/>
        </xs:choice>
        <!-- all data elements still have to be refined-->
    </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>
</xs:schema>
```

7.3.2 Ambient Information Module – ‘ambieninformation.xsd’

This module specifies the structure of ambient information. Several ‘AmbientData’ records are grouped to an ‘AmbientProfile’. In this way, the ‘AmbientProfile’ contains all available ambient information for the given area, described by ‘AmbientProfile.Location’. Each ‘AmbientData’ record has additionally a ‘Location’ reference, which describes the area of the particular data value, i.e. range of a sensor. Furthermore, each ‘AmbientData’ record is described by a source identifier ‘sourceID’. The concrete types of ‘AmbientData’ is defined as ‘xs:any’ to store any kind of ambient information. In real implementations, a suitable common refinement has to be defined.

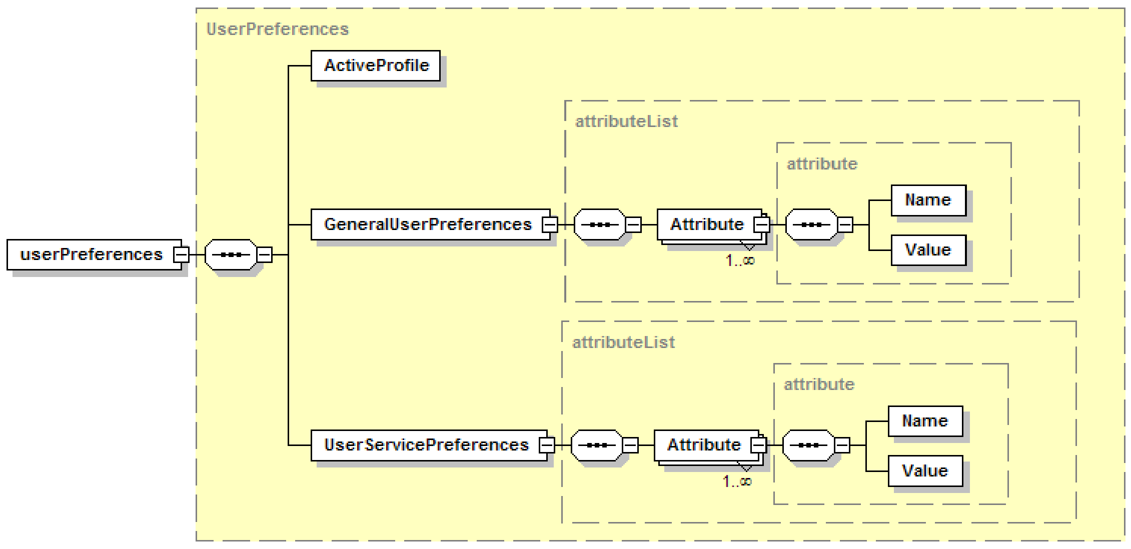


```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
 attributeFormDefault="qualified">
 <xs:include schemaLocation="geographicalposition.xsd"/>
 <xs:attributeGroup name="Common">
    <xs:attribute name="source" type="xs:string" use="optional"/>
    <xs:attribute name="quality" use="optional" default="0.5">
        <xs:simpleType>
            <xs:restriction base="xs:float">
                <xs:minInclusive value="0.0"/>
                <xs:maxInclusive value="1.0"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="volatile" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="time" type="xs:dateTime" use="required"/>
    <xs:attribute name="duration" type="xs:duration" use="optional" default="P0Y0M0DT01H30M"/>
    <xs:attribute name="ID" type="xs:string" use="optional"/>
    <!-- type="xs:anyURI" -->
 </xs:attributeGroup>
 <xs:complexType name="AmbientData">
    <xs:sequence>
        <xs:element name="Location" type="Location" minOccurs="0"/>
        <xs:element name="sourceID" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any processContents="strict" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="Common"/>
 </xs:complexType>
 <xs:complexType name="AmbientProfile">
    <xs:sequence>
        <xs:element name="Location" type="Location" minOccurs="0"/>
        <xs:element name="AmbientData" type="AmbientData" maxOccurs="unbounded"/>
    </xs:sequence>
 </xs:complexType>
```

```
<xs:element name="AmbientProfile" type="AmbientProfile"/>
</xs:schema>
```

7.3.3 Preferences Module – ‘preferences.xsd’

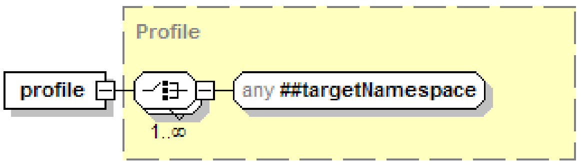
This module defines the specification of preference data for the usage in the Delivery Context. The preferences data consist of the identifier of the active User Profile as well as the attributes of the corresponding general user preferences and service-related user preferences.



```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
 attributeFormDefault="unqualified">
 <xs:complexType name="attribute">
     <xs:sequence>
         <xs:element name="Name"/>
         <xs:element name="Value"/>
     </xs:sequence>
 </xs:complexType>
 <xs:complexType name="attributeList">
     <xs:sequence>
         <xs:element name="Attribute" type="attribute" maxOccurs="unbounded"/>
     </xs:sequence>
 </xs:complexType>
 <xs:complexType name="UserPreferences">
     <xs:sequence>
         <xs:element name="ActiveProfile"/>
         <xs:element name="GeneralUserPreferences" type="attributeList"/>
         <xs:element name="UserServicePreferences" type="attributeList"/>
     </xs:sequence>
 </xs:complexType>
 <xs:element name="userPreferences" type="UserPreferences"/>
</xs:schema>
```

7.3.4 Capabilities Module – ‘xdcl/xdcl-1.0.xsd’

This module defines the specification of the capabilities of the Access Mechanism. It consists itself of further sub-modules, which are referenced by corresponding ‘xs:include’ statements.



```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="xdcl" xmlns="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
 <xs:include schemaLocation="xdcl-framework-1.0.xsd"/>
 <xs:include schemaLocation="xdcl-general-1.0.xsd"/>
 <xs:include schemaLocation="xdcl-info-1.0.xsd"/>
 <xs:include schemaLocation="xdcl-hardware-1.0.xsd"/>
 <xs:include schemaLocation="xdcl-content-1.0.xsd"/>
 <xs:include schemaLocation="xdcl-text-1.0.xsd"/>
 <xs:include schemaLocation="xdcl-input-1.0.xsd"/>
```

```
<xs:include schemaLocation="xdcl-html-1.0.xsd"/>
<xs:include schemaLocation="xdcl-graphics-1.0.xsd"/>
<xs:include schemaLocation="xdcl-sound-1.0.xsd"/>
<xs:include schemaLocation="xdcl-network-1.0.xsd"/>
<xs:include schemaLocation="xdcl-bluetooth-1.0.xsd"/>
<xs:include schemaLocation="xdcl-wap-1.0.xsd"/>
<xs:include schemaLocation="xdcl-push-1.0.xsd"/>
<xs:include schemaLocation="xdcl-mms-1.0.xsd"/>
<xs:include schemaLocation="xdcl-java-1.0.xsd"/>
<xs:include schemaLocation="xdcl-mexe-1.0.xsd"/>
<!-- ********** core ********** -->
<xs:complexType name="Profile">
   <xs:choice maxOccurs="unbounded">
      <xs:any namespace="##targetNamespace"/>
   </xs:choice>
   <xs:attribute name="ID" type="xs:string" use="optional"/>
</xs:complexType>
<xs:element name="profile" type="Profile"/>
</xs:schema>
```

The individual sub-modules are presented in the following sections.

7.3.4.1 ‘xdcl-framework-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
<xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
<xs:element name="include">
   <xs:complexType>
      <xs:attributeGroup ref="Common"/>
      <xs:attribute name="uri" type="xs:anyURI" use="required"/>
   </xs:complexType>
</xs:element>
<xs:element name="group">
   <xs:complexType>
      <xs:choice maxOccurs="unbounded">
         <xs:any namespace="##targetNamespace"/>
      </xs:choice>
      <xs:attributeGroup ref="Common"/>
   </xs:complexType>
</xs:element>
</xs:schema>
```

7.3.4.2 ‘xdcl-general-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="xdcl" xmlns="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
<xs:include schemaLocation="xdcl-framework-1.0.xsd"/>
<xs:element name="UserAgent">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="CapabilityClass">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
</xs:schema>
```

7.3.4.3 ‘xdcl-info-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
<xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
<xs:element name="Name">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="Vendor">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="Version">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="VersionType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
</xs:schema>
```

7.3.4.4 ‘xdcl-hardware-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
<xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
<xs:element name="CPU" type="xs:string"/>
<xs:element name="MemorySize" type="xs:unsignedLong">
   <xs:annotation>
      <xs:documentation>in kilobytes</xs:documentation>
   </xs:annotation>
</xs:element>
</xs:schema>
```

7.3.4.5 ‘xdcl-content-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
<xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
<xs:simpleType name="allMimeTypes">
   <xs:restriction base="xs:string">
      <xs:enumeration value="application/vnd.wap.wmlc"/>
      <xs:enumeration value="application/vnd.wap.wmlscriptc"/>
      <xs:enumeration value="image/gif"/>
      <xs:enumeration value="image/jpeg"/>
      <xs:enumeration value="image/png"/>
      <xs:enumeration value="image/tiff"/>
      <xs:enumeration value="image/vnd.wap.wbmp"/>
      <xs:enumeration value="application/x-msdos-exe"/>
      <xs:enumeration value="application/vnd.wap.wtls-ca-certificate"/>
      <xs:enumeration value="text/plain"/>
   </xs:restriction>
</xs:simpleType>
<xs:simpleType name="allEncodings">
   <xs:restriction base="xs:string">
      <xs:enumeration value="base64"/>
      <xs:enumeration value="quoted-printable"/>
   </xs:restriction>
</xs:simpleType>
<xs:simpleType name="allLanguages">
   <xs:restriction base="xs:string">
      <xs:enumeration value="zh-CN"/>
      <xs:enumeration value="en"/>
      <xs:enumeration value="fr"/>
   </xs:restriction>
</xs:simpleType>
<xs:element name="Accept">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="allMimeTypes">
            <xs:attribute name="size" type="xs:unsignedInt" use="optional"/>
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="Accept-Charset">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="allCharsets">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="Accept-Encoding">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="allEncodings">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="Accept-Language">
   <xs:complexType>
```

```
        <xs:simpleContent>
            <xs:extension base="allLanguages">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="MaxSize">
    <xs:annotation>
        <xs:documentation>in kilobytes</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="AcceptDownloadableSoftware">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="Bool">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="DownloadableSoftwareSupport">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="allMimeTypes">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
</xs:schema>
```

7.3.4.6 ‘xdcl-text-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
<xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
<xs:element name="ScreenHeightChar">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="ScreenWidthChar">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="StandardFontProportional">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="Bool">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="OutputCharset">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="allCharsets">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
</xs:schema>
```

7.3.4.7 ‘xdcl-input-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
<xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
<xs:element name="TextInputCapable">
    <xs:complexType>
        <xs:simpleContent>
```

```
         <xs:extension base="Bool">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="InputCharset">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="allCharsets">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:simpleType name="KeyboardType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="Disambiguating"/>
      <xs:enumeration value="PhoneKeypad"/>
      <xs:enumeration value="Qwerty"/>
   </xs:restriction>
</xs:simpleType>
<xs:element name="Keyboard">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="KeyboardType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="NumberOfSoftkeys">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedByte">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:simpleType name="PointingResolutionType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="Character"/>
      <xs:enumeration value="Line"/>
      <xs:enumeration value="Pixel"/>
   </xs:restriction>
</xs:simpleType>
<xs:element name="PointingResolution">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="PointingResolutionType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="VoiceInputCapable">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="Bool">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:simpleType name="AudioInputEncoderType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="G.711"/>
      <xs:enumeration value="G.931"/>
      <xs:enumeration value="mp3"/>
   </xs:restriction>
</xs:simpleType>
<xs:element name="AudioInputEncoder">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="AudioInputEncoderType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:simpleType name="VideoInputEncoderType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="MPEG-1"/>
      <xs:enumeration value="MPEG-1"/>
      <xs:enumeration value="H.261"/>
   </xs:restriction>
</xs:simpleType>
<xs:element name="VideoInputEncoder">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="VideoInputEncoderType">
               <xs:attributeGroup ref="Common"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

7.3.4.8 'xdcl-html-1.0.xsd'

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
 <xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
 <xs:element name="HtmlVersion">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="VersionType">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:element name="XhtmlVersion">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="VersionType">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:simpleType name="XhtmlModulesType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="XHTML1-struct"/>
        <xs:enumeration value="XHTML1-frames"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:element name="XhtmlModules">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="XhtmlModulesType">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:element name="JavaScriptEnabled">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="Bool">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:element name="JavaScriptVersion">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="VersionType">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:element name="TablesCapable">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="Bool">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:element name="FramesCapable">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="Bool">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
</xs:schema>
```

7.3.4.9 'xdcl-graphics-1.0.xsd'

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
 <xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
 <xs:element name="DisplayWidth">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:element name="DisplayHeight">
```

```
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedInt">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="BitsPerPixel">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedByte">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:simpleType name="ColorType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="binary"/>
      <xs:enumeration value="grey"/>
      <xs:enumeration value="limited"/>
      <xs:enumeration value="mapped"/>
      <xs:enumeration value="full"/>
   </xs:restriction>
</xs:simpleType>
<xs:element name="Color">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="ColorType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="ImageCapable">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="Bool">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:simpleType name="SVGmodulesType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="SVG tiny"/>
      <xs:enumeration value="SVG basic"/>
   </xs:restriction>
</xs:simpleType>
<xs:element name="SVGmodules">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="SVGmodulesType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="Xdpi">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedInt">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="Ydpi">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedInt">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="FontSmoothingEnabled">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="Bool">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   </xs:element>
   <xs:element name="UpdateInterval">
      <xs:complexType>
         <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
               <xs:attributeGroup ref="Common"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

7.3.4.10 'xdcl-sound-1.0.xsd'

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
 <xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
 <xs:element name="SoundOutputCapable">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="Bool">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:element name="Quality">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:element name="Speech">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="Bool">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:simpleType name="SpeakerType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="male"/>
        <xs:enumeration value="female"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:element name="Speaker">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="SpeakerType">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
</xs:schema>
```

7.3.4.11 'xdcl-network-1.0.xsd'

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
 <xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
 <xs:simpleType name="IPtype">
    <xs:restriction base="xs:string"/>
 </xs:simpleType>
 <xs:element name="IPaddress">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="IPtype">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:simpleType name="BearerType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="CSD"/>
        <xs:enumeration value="GPRS"/>
        <xs:enumeration value="GSM1800"/>
        <xs:enumeration value="GSM900"/>
        <xs:enumeration value="HSCSD"/>
        <xs:enumeration value="SMS"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:element name="SupportedBearer">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="BearerType">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:element name="CurrentBearer">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="BearerType">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
```

```
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:simpleType name="SecuritySupportType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="WTLS-1"/>
      <xs:enumeration value="WTLS-2"/>
      <xs:enumeration value="WTLS-3"/>
      <xs:enumeration value="signText"/>
      <xs:enumeration value="PPTP"/>
   </xs:restriction>
</xs:simpleType>
<xs:element name="SecuritySupport">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="SecuritySupportType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="BandwidthIn">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedLong">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="BandwidthInList">
   <xs:complexType>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:element ref="BandwidthIn" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="BandwidthOut">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedLong">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="ThruputIn">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedLong">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="ThruputOut">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedLong">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="ThruputStdDevIn">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedInt">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="ThruputStdDevOut">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedInt">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="RndtripIn">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedInt">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="RndtripOut">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:unsignedInt">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

```
<xs:element name="RndtripStdDevIn">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="RndtripStdDevOut">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="DelayIn">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="DelayOut">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="DelayStdDevIn">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="DelayStdDevOut">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="BitErrorsIn">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="BitErrorsOut">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="FrameErrorRateIn">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="FrameErrorRateOut">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="OmissionRateIn">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="OmissionRateOut">
```

```
      <xs:complexType>
         <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
               <xs:attributeGroup ref="Common"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
   <xs:element name="ConnSetupDelayIn">
      <xs:complexType>
         <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
               <xs:attributeGroup ref="Common"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
   <xs:element name="ConnSetupDelayOut">
      <xs:complexType>
         <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
               <xs:attributeGroup ref="Common"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
   <xs:element name="ConnSetupFailIn">
      <xs:complexType>
         <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
               <xs:attributeGroup ref="Common"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
   <xs:element name="ConnSetupFailOut">
      <xs:complexType>
         <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
               <xs:attributeGroup ref="Common"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
   <xs:element name="MeanUptime">
      <xs:complexType>
         <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
               <xs:attributeGroup ref="Common"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

7.3.4.12 ‘xdcl-bluetooth-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
 <xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
 <xs:element name="SupportedBluetoothVersion">
    <xs:complexType>
       <xs:simpleContent>
          <xs:extension base="VersionType">
             <xs:attributeGroup ref="Common"/>
          </xs:extension>
       </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:simpleType name="BluetoothProfilesType">
    <xs:restriction base="xs:string">
       <xs:enumeration value="dialup"/>
       <xs:enumeration value="lanAccess"/>
    </xs:restriction>
 </xs:simpleType>
 <xs:element name="BluetoothProfiles">
    <xs:complexType>
       <xs:simpleContent>
          <xs:extension base="BluetoothProfilesType">
             <xs:attributeGroup ref="Common"/>
          </xs:extension>
       </xs:simpleContent>
    </xs:complexType>
 </xs:element>
</xs:schema>
```

7.3.4.13 ‘xdcl-wap-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
```

```
<xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
<xs:element name="WapVersion">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="VersionType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="WmlVersion">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="VersionType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="WapDeviceClass">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="WmlScriptVersion">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="VersionType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:simpleType name="WmlScriptLibraryType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="Lang"/>
      <xs:enumeration value="Float"/>
      <xs:enumeration value="Literal"/>
      <xs:enumeration value="URL"/>
      <xs:enumeration value="WMLBrowser"/>
      <xs:enumeration value="Dialogs"/>
   </xs:restriction>
</xs:simpleType>
<xs:element name="WmlScriptLibrary">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="WmlScriptLibraryType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:element name="WtaVersion">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="VersionType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:simpleType name="WtaiLibraryType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="WTAPublic.addPBEntry"/>
      <xs:enumeration value="WTAPublic.makeCall"/>
      <xs:enumeration value="WTAPublic.sendDTMF"/>
   </xs:restriction>
</xs:simpleType>
<xs:element name="WtaiLibrary">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="WtaiLibraryType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:simpleType name="DrmClassType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="ForwardLock"/>
      <xs:enumeration value="CombinedDelivery"/>
      <xs:enumeration value="SeparateDelivery"/>
   </xs:restriction>
</xs:simpleType>
<xs:element name="DrmClass">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="DrmClassType">
            <xs:attributeGroup ref="Common"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
<xs:simpleType name="DrmConstraintsType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="datetime"/>
```

```
         <xs:enumeration value="interval"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:element name="DrmConstraints">
      <xs:complexType>
         <xs:simpleContent>
            <xs:extension base="DrmConstraintsType">
               <xs:attributeGroup ref="Common"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
   <xs:element name="OmaDownload">
      <xs:complexType>
         <xs:simpleContent>
            <xs:extension base="Bool">
               <xs:attributeGroup ref="Common"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
   <xs:simpleType name="SupportedPictogramSetType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="core"/>
         <xs:enumeration value="core/operation"/>
         <xs:enumeration value="human"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:element name="SupportedPictogramSet">
      <xs:complexType>
         <xs:simpleContent>
            <xs:extension base="SupportedPictogramSetType">
               <xs:attributeGroup ref="Common"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

7.3.4.14 ‘xdcl-push-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
 <xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
 <xs:element name="MaxPushReq">
    <xs:complexType>
       <xs:simpleContent>
          <xs:extension base="xs:unsignedInt">
             <xs:attributeGroup ref="Common"/>
          </xs:extension>
       </xs:simpleContent>
    </xs:complexType>
 </xs:element>
</xs:schema>
```

7.3.4.15 ‘xdcl-mms-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
 <xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
 <xs:element name="MmsVersion">
    <xs:complexType>
       <xs:simpleContent>
          <xs:extension base="VersionType">
             <xs:attributeGroup ref="Common"/>
          </xs:extension>
       </xs:simpleContent>
    </xs:complexType>
 </xs:element>
 <xs:element name="MmsStreamingCapable">
    <xs:complexType>
       <xs:attributeGroup ref="Common"/>
    </xs:complexType>
 </xs:element>
</xs:schema>
```

7.3.4.16 ‘xdcl-java-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
 <xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
 <xs:element name="JavaEnabled">
    <xs:complexType>
       <xs:simpleContent>
          <xs:extension base="Bool">
             <xs:attributeGroup ref="Common"/>
```

```
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="JavaAppletEnabled">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="Bool">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:simpleType name="JVMVersionType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="SunJRE1.2"/>
        <xs:enumeration value="MSJVM1.0"/>
    </xs:restriction>
</xs:simpleType>
<xs:element name="JVMVersion">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="JVMVersionType">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:simpleType name="JavaPlatformType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="PersonalJava"/>
        <xs:enumeration value="CLDC"/>
        <xs:enumeration value="MIDP"/>
    </xs:restriction>
</xs:simpleType>
<xs:element name="JavaPlatform">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="JavaPlatformType">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="JavaPackage">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="JavaProtocol">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
</xs:schema>
```

7.3.4.17 ‘xdcl-mexe-1.0.xsd’

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by KJ (KJ) -->
<xs:schema targetNamespace="xdcl" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="xdcl"
 elementFormDefault="unqualified" attributeFormDefault="unqualified" blockDefault="#all"
 finalDefault="#all">
<xs:include schemaLocation="xdcl-attribstypes-1.0.xsd"/>
<xs:element name="MexeClassmark">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="MexeSpec">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="MexeSecureDomains">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="Bool">
                <xs:attributeGroup ref="Common"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
```

```
31: </xs:element>
32: </xs:schema>
```

7.3.5 Overview of proposed XDCL Modules

This section describes the XDCL modules, which are identified in section 3.5.6.10 in more detail. Each module is characterized by a set of attributes.

7.3.5.1 General Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
UserAgent	String	No	"SonyEricssonP800/R101 Profile/MIDP-1.0 Configuration/CLDC-1.0", "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"	The user agent string as received in the HTTP header
CapabilityClasses	Set of String	No	"pushable", "PDA", "small-screen", "low-bandwidth"	The capabilities classes the device belongs to

7.3.5.2 Info Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
Name	String	No	"Internet Explorer", " 7110",	The name of a source entity
Vendor	String	No	"Microsoft", "Nokia"	The vendor of a source entity
Version	Version	No	"6.0", "4.94"	The version of a source entity

7.3.5.3 Hardware Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
CPU	String	No	"Pentium III", "PowerPC 750"	The processor used in the device
MemorySize	Number	No	32768	The available memory size in kilobytes

7.3.5.4 Content Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
CcppAccept	Set of String	No	"text/html", "text/plain", "image/gif"	List of content types the device supports
CcppAccept-Charset	Set of String	No	"US-ASCII", "ISO-8859-1", "Shift_JIS"	List of character sets the device supports
CcppAccept-Encoding	Set of String	No	"base64", "quoted-printable"	List of transfer encodings the device supports
CcppAccept-Language	Set of String	No	"zh-CN", "en", "fr"	List of preferred document languages
MaxSize	Number	No	"20480"	The maximum size of a multimedia message in bytes
AcceptDownloadableSoftware	Boolean	No	"Yes", "No"	Indicates the user's preference on whether to accept downloadable software
DownloadableSoftwareSupport	Set of String	No	"application/x-msdos-exe"	List of executable content types which the device supports and which it is willing to accept from the network

7.3.5.5 Text Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
ScreenHeightChar	Number	Yes	"4", "8"	Height of the text display area in units of characters
ScreenWidthChar	Number	Yes	"12", "16"	Width of the text display area in units of characters
StandardFontProportional	Boolean	No	"Yes", "No"	Indicates if the standard font used for the text is display is proportional
OutputCharSet	Set of String	No	"US-ASCII", "ISO-8859-1", "Shift_JIS"	The character set used for text output

7.3.5.6 Input Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
TextInputCapable	Boolean	No	"Yes", "No"	Indicates whether the device supports alpha-numeric text entry
InputCharSet	Set of String	No	"US-ASCII", "ISO-8859-1", "Shift_JIS"	List of character sets supported by the device for text entry
Keyboard	String	No	"Disambiguating", "Qwerty", "Phon-eKeypad"	Type of keyboard supported by the device
NumberOfSoftKeys	Number	No	"3", "2"	Number of soft keys available on the device
PointingResolution	String	No	"Character", "Line", "Pixel"	Type of resolution of the pointing acces-sory supported by the device
VoiceInputCapable	Boolean	No	"Yes", "No"	Indicates whether the device supports any form of voice input, including speech recognition
AudioInputEncoder	Set of String	No	"G.711", "G.931"	List of audio input encoders supported by the device
VideoInputEncoder	Set of String	No	"MPEG-1", "MPEG-2", "H.261"	List of video input encoders supported by the device

7.3.5.7 HTML Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
HtmlVersion	Version	No	"2.0", "3.2", "4.0"	HTML version supported by the browser
XhtmlVersion	Version	No	"1.0"	XHTML version supported by the browser
XhtmlModules	Set of String	No	"XHTML1-struct", "XHTML1-frames"	List of XHTML modules supported by the browser
JavaScriptEnabled	Boolean	No	"Yes", "No"	Indicates whether the browser supports JavaScript
JavaScriptVersion	Version	No	"1.4"	Version of the JavaScript language sup-ported by the browser
TablesCapable	Boolean	No	"Yes", "No"	Indicates whether the browser is capable of displaying HTML tables
FramesCapable	Boolean	No	"Yes", "No"	Indicates whether the browser is capable of displaying HTML frames

7.3.5.8 Graphics Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
DisplayWidth	Number	Yes	"160", "640"	The width of the display area in units of pixels

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
DisplayHeight	Number	Yes	"160", "480"	The height of the display area in units of pixels
BitsPerPixel	Number	No	"2", "8"	The number of bits of colour or grey-scale information per pixel
Color	String	No	"binary", "grey", "limited", "mapped", "full"	colour capabilities – see RFC 2534
ImageCapable	Boolean	No	"Yes", "No"	Whether the device supports the display of images
SVGmodules	Set of Strings	No	"SVG tiny", "SVG basic"	
Xdpi	Number	No		the number of horizontal dots per inch (DPI) of the system's screen
Ydpi	Number	No		the number of vertical dots per inch (DPI) of the system's screen
fontSmoothingEnabled	Boolean	No		whether the user has enabled font smoothing in the Display control panel
updateInterval	Number	No		the update interval for the screen (frame rate)

7.3.5.9 Sound Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
SoundOutputCapable	Boolean	No	"Yes", "No"	Indicates whether the device supports sound output
Quality	String	No		Describes the possible sound output quality (TBD)
Speech	Boolean	No	"Yes", "No"	Whether the device has a speech synthesis engine
SpeechSpeaker	String	No	"Male", "Female"	The preference for the synthetic speaker

7.3.5.10 Network Module

The ‘Network’ module shall provide an example of how a module could look like in its full verbosity. The vocabulary has been united with the parameters defined by the Quality of Service (QoS) definition and contains a possibly very detailed description of the network properties. All the modules could have a similar verbosity in their final specification by respective interest groups.

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
IPaddress	String	Yes	"168.0.0.1"	The current IP address of the client device
SupportedBearers	Set of String	Yes	"GPRS", "GUTS", "TwowaySMS", CSD", "USSD"	List of bearers supported by the device (WAP)
CurrBearerService	String	Yes	"OneWaySMS", "GUTS", "TwoWay-Packet"	The bearer on which the current session was opened (WAP)
SecuritySupport	String	Yes	"WTLS-1", WTLS-2", "WTLS-3", "signText", "PPTP"	Type of security or encryption mechanism supported (WAP)
BandwidthIn	Number	Yes		The bandwidth in kBits/s in the inbound direction

Attribute	***Data type***	***Vol.***	***Examples***	***Description***
BandwidthOut	Number	Yes		The bandwidth in kBits/s in the outbound direction
ThruputIn	Number	Yes		The number of user data bits in kBits/s successfully transferred in the inbound direction
ThruputOut	Number	Yes		The number of user data bits in kBits/s successfully transferred in the outbound direction
ThruputStdDevIn	Number	Yes		The current standard deviation of the throughput within a time unit (kBits/s)
ThruputStdDevOut	Number	Yes		The current standard deviation of the throughput within a time unit (kBits/s)
RndtripIn	Number	Yes		The round trip time in ms
RndtripOut	Number	Yes		The round trip time in ms
RndtripStdDevIn	Number	Yes		The standard deviation of the round-trip time within a time unit (ms)
RndtripStdDevOut	Number	Yes		The standard deviation of the round-trip time within a time unit (ms)
DelayIn	Number	Yes		The (nominal) time required for a data segment to be transmitted to a peer entity (ms)
DelayOut	Number	Yes		The (nominal) time required for a data segment to be transmitted to a peer entity (ms)
DelayStdDevIn	Number	Yes		The standard deviation of the delay time within a time unit (ms)
DelayStdDevOut	Number	Yes		The standard deviation of the delay time within a time unit (ms)
BitErrorsIn	Number	Yes		The ratio of the number of bit errors to the total number of bits transmitted in a given time interval.
BitErrorsOut	Number	Yes		The ratio of the number of bit errors to the total number of bits transmitted in a given time interval.
FrameErrorRateIn	Number	Yes		The probability that a data segment is not transmitted correctly over a link.
FrameErrorRateOut	Number	Yes		The probability that a data segment is not transmitted correctly
OmissionRateIn	Number	Yes		The probability that a data segment is not transmitted correctly between 0 and 1
OmissionRateOut	Number	Yes		The probability that a data segment is not transmitted correctly between 0 and 1
ConnSetupDelayIn	Number	Yes		The (sampled) delay to establish a connection (ms)
ConnSetupDelayOut	Number	Yes		The (sampled) delay to establish a connection (ms)
ConnSetupFailIn	Number	Yes		The ratio of total call attempts that result in call setup failure to the total call attempts in a population of interest
ConnSetupFailOut	Number	Yes		The ratio of total call attempts that result in call setup failure to the total call attempts in a population of interest
MeanUptime	Number	Yes		The expected uptime of an established link (s)

7.3.5.11 Bluetooth Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
SupportedBluetoothVersion	Version	No	"1.0"	Supported Bluetooth version
BluetoothProfiles	Set of String	No	"dialup", "lanAccess"	Supported Bluetooth profiles as defined in the Bluetooth specification

7.3.5.12 WAP Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
WmlVersion	Set of Version	No	"1.1", "2.0"	List of WML language versions supported by the device
WapDeviceClass	String	No	"A"	Classification of the device based on capabilities as identified in the WAP 1.1 specifications
WmlScriptVersion	Set of Version	No	"1.1", "1.2"	List of WMLScript version numbers supported by the device
WmlScriptLibraries	Set of String	No	"Lang", "Float", "Literal", "URL", "WMLBrowser", "Dialogs"	List of mandatory and optional libraries supported in the device's WMLScript VM
WtaVersion	Version	No	"1.1"	Version of WTA user agent
WtaiLibraries	Set of String	No	"WTAVoiceCall", "WTANetText", "WTAPhoneBook", "WTACallLog", "WTAMisc", "WTAGSM", "WTAIS136", "WTAPDC"	List of WTAI network common and network specific libraries supported by the device.
DrmClass	Set of String	No	"ForwardLock", "CombinedDelivery", "SeparateDelivery"	List of DRM capabilities
DrmConstraints	Set of String	No	"datetime", "interval"	List of optional DRM permission constraints
OmaDownload	Boolean	No	"Yes", "No"	Supports OMA Download
SupportedPictogramSet	Set of String	No	"core", "core/operation", "human"	Pictogram classes supported by the device as defined in the WAP Pictogram specification

7.3.5.13 Push Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
Push-Accept-AppID	Set of String	No	"x-wapapplication:wml.ua", "*"	List of applications the device supports for push
Push-MaxPushReq	Number	No	"1", "5"	Maximum number of outstanding push requests that the device can handle

7.3.5.14 MMS Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
MmsVersion	Set of Version	No	"2.0", "1.3"	The MMS versions supported by the MMS Client conveyed as majorVersionNumber. minorVersionNumber.
MmsCcppStreamingCapable	Boolean	No	"Yes", "No"	Indicates whether the MMS Client is capable of invoking streaming.

7.3.5.15 Java Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
JavaEnabled	Boolean	No	"Yes", "No"	Indicates whether the device supports a Java virtual machine
JavaAppletEnabled	Boolean	No	"Yes", "No"	Indicates whether the browser supports Java applets
JVMVersion	Set of String	No	"SunJRE1.2", "MSJVM1.0"	List of the Java virtual machines installed on the device
JavaPlatform	Set of String	No	"PersonalJava", "CLDC", "MIDP"	The list of JAVA platforms and profiles installed in the device.
JavaPackage	Set of String	No	"com.acme.regexp/1.1", "com.acme.helper/3.0"	Lists the optional packages installed on the device over and above those that are part of the Java profile
JavaProtocol	Set of String	No	"sms/1.0", "file/1.0"	Lists the protocols supported by the device above those that are part of the standard Java profile indicated

7.3.5.16 MExE Module

Attribute	*Data type*	*Vol.*	*Examples*	*Description*
MexeClassmarks	Set of String	No	"1", "2"	ETSI MExE classmark
MexeSpec	String	No	"7.02"	Class mark specialization
MexeSecureDomains	Boolean	No	"Yes", "No"	Indicates whether the device supports MExE security domains as specified in the MExE specifications

7.4 Sample Documents

7.4.1 Generic User Interface described in GUIML

```
<document xmlns:xfm=”http://www.w3.org/2002/01/xforms”
          xmlns:smil=”http://www.w3.org/2001/SMIL20/Language”
          xmlns:xlink=”http://www.w3.org/1999/xlink”>
 <head>
    <title>Weather Service</title>
 </head>
 <body>
    <frame href=”weather-menubar” />
    <frame id=”main” title=”Weather”>
       <h1>Weather Forecast</h1>
       <h2>Berlin, Sunday, 29 June 2003</h2>
       <smil:switch>
          <smil:ref type=”JPEG”
             src=”http://weather.com/forecast/images/weather-brd.jpg”/>
          <smil:ref type=”text” language=”en”
             src=”http://weather.com/forecast/weathertext-brd.txt” />
          <smil:ref type=”text” language=”de”
             src=”http://weather.com/forecast/weathertext-brd-de.txt” />
       </smil:switch>
       <xfm:model id="weatherservice">
          <xfm:submission action="/OKSPortalWeatherService/Servlet"
                           method="get"/>
          <xfm:instance>
             <zipCode/>
          </xfm:instance>
       </xfm:model>
          <xfm:input model="weatherservice" ref="zipCode">
             <xfm:label>
                <smil:switch>
                   <smil:ref src="data:Zip Code: " language="en"/>
                   <smil:ref src="data:Postleitzahl: "
                         language="de"/>
                </smil:switch>
             </xfm:label>
          </xfm:input>
          <xfm:submit model="weatherservice">
             <xfm:label>
                <smil:switch>
                   <smil:ref src="data:Submit" language="en"/>
                   <smil:ref src="data:Senden" language="de"/>
                </smil:switch>
             </xfm:label>
          </xfm:submit>
    </frame>
    </body>
</document>
```

7.4.2 XForms Usage Example

```
<?xml version="1.0" encoding="UTF-8"?>
<document xmlns:xfm="http://www.w3.org/2002/01/xforms"
 xmlns:smil="http://www.w3.org/2001/SMIL20/Language" xmlns:xlink="http://www.w3.org/1999/xlink">
 <xfm:group ref="selectservice">
    <xfm:selectOne ref="@service">
       <xfm:caption>Select Service</xfm:caption>
       <xfm:choices>
          <xfm:item>
             <xfm:caption>Weather</xfm:caption>
             <xfm:value>weather</xfm:value>
          </xfm:item>
          <xfm:item>
             <xfm:caption>Home</xfm:caption>
             <xfm:value>home</xfm:value>
          </xfm:item>
       </xfm:choices>
    </xfm:selectOne>
    <xfm:input ref="login">
       <xfm:caption>login</xfm:caption>
    </xfm:input>
    <xfm:submit submitInfo="s00">
       <xfm:caption>Submit Form</xfm:caption>
    </xfm:submit>
</xfm:group>
</document>
```

7.4.3 Capabilities Module Example

```
<profile xmlns:screen="http://www.example.org/screen-module#"
 xmlns:net="http://www.example.org/network-module#">
 <net:transferrate source="HW-79686789" q="0.0">512</net:transferrate>
 <screen:screenwidth source="HW-79686789" q="0.0">800</screen:screenwidth>
 <screen:screenheight source="HW-79686789" q="0.0">600</screen:screenheight>
 <screen:imagecapable source="HW-79686789" q="0.0">Yes</screen:imagecapable>
 <net:transferrate source="BRWS-5748923075" q="0.7">128</net:transferrate>
 <net:transferrate q="0.9">64</net:transferrate>
</profile>
```

7.4.4 AmbientProfile Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AmbientProfile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="ai
 ambient.xsd weather weather.xsd">
 <AmbientData time="2003-06-24T09:30:47+01:00">
    <location>
       <longitude>13.18831E</longitude>
       <latitude>52.31555N</latitude>
       <uncertaintyShape>
          <circle unit="meter">23</circle>
       </uncertaintyShape>
    </location>
</AmbientData>
<AmbientData quality="0.9" ID="D56FFAA8-6D13-11d4-B675-0010F3008057" source="TEMP456" time="2003-
 06-24T09:30:47+01:00">
    <weather:temperature unit="celcius">31</weather:temperature>
</AmbientData>
<AmbientData quality="0.8" ID="2C933718-CF6A-4f70-AB37-9F780B427709" source="HUM987" time="2003-
 06-24T09:35:32+01:00">
    <weather:humidity>81</weather:humidity>
</AmbientData>
<AmbientData quality="0.4" ID="719C511C-4C40-48fe-9ACF-1224B1DF0C0E" source="weatherInterpreter"
 time="2003-06-24T09:35:00+01:00">
    <sourceID>D56FFAA8-6D13-11d4-B675-0010F3008057</sourceID>
    <sourceID>2C933718-CF6A-4f70-AB37-9F780B427709</sourceID>
    <weather:weather>muggy</weather:weather>
</AmbientData>
</AmbientProfile>
```

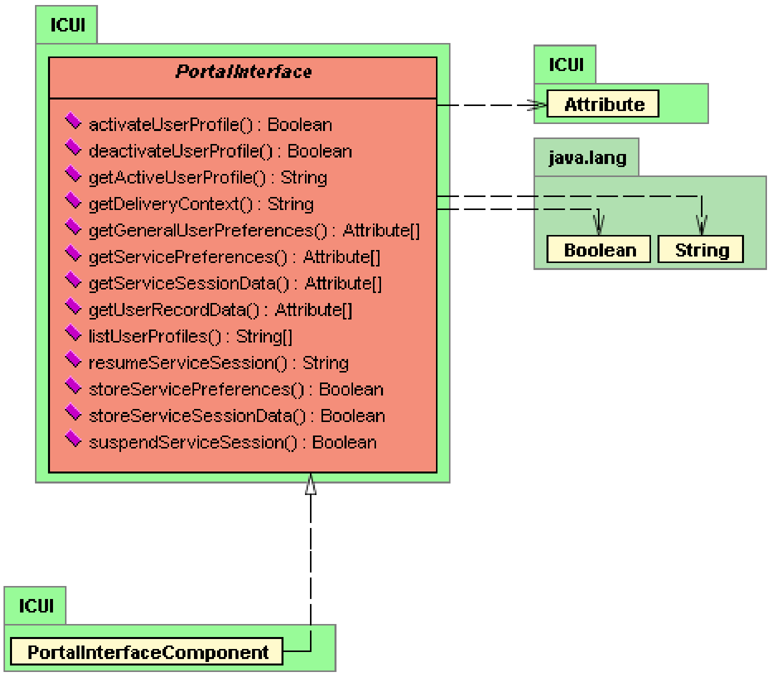
7.4.5 Complete Delivery Context Example

```
<?xml version="1.0" encoding="UTF-8"?>
<DeliveryContext xmlns:xdcl="xdcl" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="deliverycontext.xsd">
 <UserLocation>
    <longitude>13.18831E</longitude>
    <latitude>52.31555N</latitude>
    <uncertaintyShape>
       <circle unit="meter">45</circle>
    </uncertaintyShape>
 </UserLocation>
<User>steglich@cs.tu-berlin.de</User>
<Capabilities ID="B2CDAFA9A92-22D1-C375-1020C5424143"/>
<AmbientInformation>
    <Location>
       <longitude>13.18831E</longitude>
       <latitude>52.31555N</latitude>
```

```
      <uncertaintyShape>
         <circle unit="meter">5</circle>
      </uncertaintyShape>
   </Location>
   <AmbientData source="TEMP456" quality="0.9" volatile="false" time="2003-06-24T09:30:47+01:00" duration="P0Y0M0DT01H30M" ID="D56FFAA8-6D13-11d4-B675-0010F3008057">
      <Location>
         <longitude>13.18831E</longitude>
         <latitude>52.31555N</latitude>
         <uncertaintyShape>
            <circle unit="meter">10</circle>
         </uncertaintyShape>
      </Location>
      <weather:temperature unit="celcius">31</weather:temperature>
   </AmbientData>
   <AmbientData source="HUM987" quality="0.8" volatile="false" time="2003-06-24T09:35:32+01:00" duration="P0Y0M0DT02H00M" ID="2C933718-CF6A-4f70-AB37-9F780B427709">
      <Location>
         <longitude>13.18831E</longitude>
         <latitude>52.31555N</latitude>
         <uncertaintyShape>
            <circle unit="meter">15</circle>
         </uncertaintyShape>
      </Location>
      <weather:humidity>81</weather:humidity>
   </AmbientData>
   <AmbientData quality="0.4" ID="719C511C-4C40-48fe-9ACF-1224B1DF0C0E" source="weatherInterpreter" time="2003-06-24T09:36:00+01:00">
      <sourceID>D56FFAA8-6D13-11d4-B675-0010F3008057</sourceID>
      <sourceID>2C933718-CF6A-4f70-AB37-9F780B427709</sourceID>
      <weather:weather>muggy</weather:weather>
   </AmbientData>
</AmbientInformation>
<Preferences>
   <ActiveProfile>Office</ActiveProfile>
   <GeneralUserPreferences>
      <Attribute>
         <Name>PreferredLanguage</Name>
         <Value>English</Value>
      </Attribute>
      <Attribute>
         <Name>PreferredTerminal</Name>
         <Value>sts.home.cs.tu-berlin.de</Value>
      </Attribute>
   </GeneralUserPreferences>
   <UserServicePreferences>
      <Attribute>
         <Name>NewsCategories</Name>
         <Value>Business, Sports</Value>
      </Attribute>
      <Attribute>
         <Name>Delivery</Name>
         <Value>sip:steglich@iptel.org smtp:steglich@cs.tu-berlin.de</Value>
      </Attribute>
   </UserServicePreferences>
</Preferences>
</DeliveryContext>
```

7.5 *I-centric User Interaction System APIs*

7.5.1 Portal Interface WSDL Specification


ICUI
PortalInterface
activateUserProfile() : Boolean
deactivateUserProfile() : Boolean
getActiveUserProfile() : String
getDeliveryContext() : String
getGeneralUserPreferences() : Attribute[]
getServicePreferences() : Attribute[]
getServiceSessionData() : Attribute[]
getUserRecordData() : Attribute[]
listUserProfiles() : String[]
resumeServiceSession() : String
storeServicePreferences() : Boolean
storeServiceSessionData() : Boolean
suspendServiceSession() : Boolean
ICUI
Attribute
java.lang
Boolean
String
ICUI
PortalInterfaceComponent

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://ICUI" xmlns="http://schemas.xmlsoap.org/wsdl/"
 xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://ICUI-impl"
 xmlns:intf="http://ICUI" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <wsdl:types>
  <schema targetNamespace="http://ICUI" xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="ArrayOfAttribute">
    <complexContent>
     <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="null]"/>
     </restriction>
    </complexContent>
   </complexType>
   <element name="ArrayOfAttribute" nillable="true" type="intf:ArrayOfAttribute"/>
   <complexType name="ArrayOf_xsd_string">
    <complexContent>
     <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string]"/>
     </restriction>
    </complexContent>
   </complexType>
   <element name="ArrayOf_xsd_string" nillable="true" type="intf:ArrayOf_xsd_string"/>
  </schema>
 </wsdl:types>

   <wsdl:message name="getServiceSessionDataRequest">
      <wsdl:part name="serviceSessionIdentifier" type="xsd:string"/>
   </wsdl:message>
   <wsdl:message name="getServicePreferencesResponse">
      <wsdl:part name="getServicePreferencesReturn" type="intf:ArrayOfAttribute"/>
   </wsdl:message>
   <wsdl:message name="activateUserProfileRequest">
      <wsdl:part name="serviceSessionIdentifier" type="xsd:string"/>
      <wsdl:part name="userProfileName" type="xsd:string"/>
   </wsdl:message>
   <wsdl:message name="resumeServiceSessionResponse">
      <wsdl:part name="resumeServiceSessionReturn" type="xsd:string"/>
   </wsdl:message>
   <wsdl:message name="getGeneralUserPreferencesResponse">
      <wsdl:part name="getGeneralUserPreferencesReturn" type="intf:ArrayOfAttribute"/>
   </wsdl:message>
   <wsdl:message name="listUserProfilesResponse">
      <wsdl:part name="listUserProfilesReturn" type="intf:ArrayOf_xsd_string"/>
   </wsdl:message>
   <wsdl:message name="getActiveUserProfileResponse">
      <wsdl:part name="getActiveUserProfileReturn" type="xsd:string"/>
   </wsdl:message>
  <wsdl:message name="getUserRecordDataResponse">
     <wsdl:part name="getUserRecordDataReturn" type="intf:ArrayOfAttribute"/>
  </wsdl:message>
  <wsdl:message name="getActiveUserProfileRequest">
     <wsdl:part name="serviceSessionIdentifier" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="getGeneralUserPreferencesRequest">
     <wsdl:part name="serviceSessionIdentifier" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="storeServiceSessionDataResponse">
     <wsdl:part name="storeServiceSessionDataReturn" type="xsd:boolean"/>
  </wsdl:message>
  <wsdl:message name="getUserRecordDataRequest">
     <wsdl:part name="serviceSessionIdentifier" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="activateUserProfileResponse">
     <wsdl:part name="activateUserProfileReturn" type="xsd:boolean"/>
  </wsdl:message>
  <wsdl:message name="getServicePreferencesRequest">
     <wsdl:part name="serviceSessionIdentifier" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="getDeliveryContextRequest">
     <wsdl:part name="AS" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="getServiceSessionDataResponse">
     <wsdl:part name="getServiceSessionDataReturn" type="intf:ArrayOfAttribute"/>
  </wsdl:message>
  <wsdl:message name="storeServiceSessionDataRequest">
     <wsdl:part name="serviceSessionIdentifier" type="xsd:string"/>
     <wsdl:part name="sessionData" type="intf:ArrayOfAttribute"/>
  </wsdl:message>
  <wsdl:message name="getDeliveryContextResponse">
     <wsdl:part name="getDeliveryContextReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="suspendServiceSessionRequest">
     <wsdl:part name="serviceSessionIdentifier" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="listUserProfilesRequest">
     <wsdl:part name="serviceSessionIdentifier" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="deactivateUserProfileResponse">
     <wsdl:part name="deactivateUserProfileReturn" type="xsd:boolean"/>
  </wsdl:message>
  <wsdl:message name="resumeServiceSessionRequest">
     <wsdl:part name="serviceSessionIdentifier" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="storeServicePreferencesResponse">
     <wsdl:part name="storeServicePreferencesReturn" type="xsd:boolean"/>
  </wsdl:message>
  <wsdl:message name="suspendServiceSessionResponse">
     <wsdl:part name="suspendServiceSessionReturn" type="xsd:boolean"/>
  </wsdl:message>
  <wsdl:message name="storeServicePreferencesRequest">
     <wsdl:part name="serviceSessionIdentifier" type="xsd:string"/>
     <wsdl:part name="attributeList" type="intf:ArrayOfAttribute"/>
     <wsdl:part name="userProfileName" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="deactivateUserProfileRequest">
     <wsdl:part name="serviceSessionIdentifier" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="PortalInterfaceComponent">
     <wsdl:operation name="getServiceSessionData" parameterOrder="serviceSessionIdentifier">
        <wsdl:input message="intf:getServiceSessionDataRequest"
name="getServiceSessionDataRequest"/>
        <wsdl:output message="intf:getServiceSessionDataResponse"
name="getServiceSessionDataResponse"/>
     </wsdl:operation>
     <wsdl:operation name="storeServiceSessionData" parameterOrder="serviceSessionIdentifier
sessionData">
        <wsdl:input message="intf:storeServiceSessionDataRequest"
name="storeServiceSessionDataRequest"/>
        <wsdl:output message="intf:storeServiceSessionDataResponse"
name="storeServiceSessionDataResponse"/>
     </wsdl:operation>
     <wsdl:operation name="suspendServiceSession" parameterOrder="serviceSessionIdentifier">
        <wsdl:input message="intf:suspendServiceSessionRequest"
name="suspendServiceSessionRequest"/>
        <wsdl:output message="intf:suspendServiceSessionResponse"
name="suspendServiceSessionResponse"/>
     </wsdl:operation>
     <wsdl:operation name="resumeServiceSession" parameterOrder="serviceSessionIdentifier">
        <wsdl:input message="intf:resumeServiceSessionRequest"
name="resumeServiceSessionRequest"/>
        <wsdl:output message="intf:resumeServiceSessionResponse"
name="resumeServiceSessionResponse"/>
     </wsdl:operation>
     <wsdl:operation name="getDeliveryContext" parameterOrder="AS">
        <wsdl:input message="intf:getDeliveryContextRequest" name="getDeliveryContextRequest"/>
        <wsdl:output message="intf:getDeliveryContextResponse"
name="getDeliveryContextResponse"/>
     </wsdl:operation>
     <wsdl:operation name="listUserProfiles" parameterOrder="serviceSessionIdentifier">
        <wsdl:input message="intf:listUserProfilesRequest" name="listUserProfilesRequest"/>
        <wsdl:output message="intf:listUserProfilesResponse" name="listUserProfilesResponse"/>
     </wsdl:operation>
     <wsdl:operation name="storeServicePreferences" parameterOrder="serviceSessionIdentifier
attributeList userProfileName">
        <wsdl:input message="intf:storeServicePreferencesRequest"
name="storeServicePreferencesRequest"/>
```

```
         <wsdl:output message="intf:storeServicePreferencesResponse" name="storeServicePreferencesResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getServicePreferences" parameterOrder="serviceSessionIdentifier">

         <wsdl:input message="intf:getServicePreferencesRequest" name="getServicePreferencesRequest"/>
         <wsdl:output message="intf:getServicePreferencesResponse" name="getServicePreferencesResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getGeneralUserPreferences" parameterOrder="serviceSessionIdentifier">
         <wsdl:input message="intf:getGeneralUserPreferencesRequest" name="getGeneralUserPreferencesRequest"/>
         <wsdl:output message="intf:getGeneralUserPreferencesResponse" name="getGeneralUserPreferencesResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getUserRecordData" parameterOrder="serviceSessionIdentifier">
         <wsdl:input message="intf:getUserRecordDataRequest" name="getUserRecordDataRequest"/>
         <wsdl:output message="intf:getUserRecordDataResponse" name="getUserRecordDataResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getActiveUserProfile" parameterOrder="serviceSessionIdentifier">
         <wsdl:input message="intf:getActiveUserProfileRequest" name="getActiveUserProfileRequest"/>
         <wsdl:output message="intf:getActiveUserProfileResponse" name="getActiveUserProfileResponse"/>
      </wsdl:operation>
      <wsdl:operation name="activateUserProfile" parameterOrder="serviceSessionIdentifier userProfileName">
         <wsdl:input message="intf:activateUserProfileRequest" name="activateUserProfileRequest"/>
         <wsdl:output message="intf:activateUserProfileResponse" name="activateUserProfileResponse"/>
      </wsdl:operation>
      <wsdl:operation name="deactivateUserProfile" parameterOrder="serviceSessionIdentifier">
         <wsdl:input message="intf:deactivateUserProfileRequest" name="deactivateUserProfileRequest"/>
         <wsdl:output message="intf:deactivateUserProfileResponse" name="deactivateUserProfileResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="PortalInterfaceComponentSoapBinding" type="intf:PortalInterfaceComponent">
      <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="getServiceSessionData">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getServiceSessionDataRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getServiceSessionDataResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="storeServiceSessionData">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="storeServiceSessionDataRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="storeServiceSessionDataResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="suspendServiceSession">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="suspendServiceSessionRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="suspendServiceSessionResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="resumeServiceSession">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="resumeServiceSessionRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="resumeServiceSessionResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getDeliveryContext">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getDeliveryContextRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getDeliveryContextResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="listUserProfiles">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="listUserProfilesRequest">
```

```
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="listUserProfilesResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="storeServicePreferences">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="storeServicePreferencesRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="storeServicePreferencesResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getServicePreferences">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getServicePreferencesRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getServicePreferencesResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getGeneralUserPreferences">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getGeneralUserPreferencesRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getGeneralUserPreferencesResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getUserRecordData">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getUserRecordDataRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getUserRecordDataResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getActiveUserProfile">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getActiveUserProfileRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getActiveUserProfileResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="activateUserProfile">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="activateUserProfileRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="activateUserProfileResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="deactivateUserProfile">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="deactivateUserProfileRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="deactivateUserProfileResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://ICUI" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="PortalInterfaceComponentService">
      <wsdl:port binding="intf:PortalInterfaceComponentSoapBinding" name="PortalInterfaceComponent">
         <wsdlsoap:address location="http://localhost:8080/ICUI/services/PortalInterfaceComponent"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

7.5.2 Personalization Interfaces

7.5.2.1 Internal Interface

```
package Personalization;
public interface Internal extends External {
   public resultCode createUserProfile(UserProfileName originUserProfileName, UserProfileName newUserProfileName);
   public resultCode deleteUserProfile(UserProfileName userProfileName);
   public resultCode updateUserProfile(UserProfileName userProfileName, AttributeList attributeList);
   public resultCode setSelectionContext(UserProfileName userProfileName, AttributeList conditionList);
   public AttributeList getSelectionContext(UserProfileName userProfileName);
   public resultCode setGeneralPreferences(UserProfileName userProfileName, AttributeList attributeList);
}
```

7.5.2.2 External Interface

```
package Personalization;
public interface External {
   public UserProfileList listUserProfiles();
   public resultCode storeServicePreferences(AttributeList attributeList, userProfileName userProfileName);
   public AttributeList getServicePreferences();
   public AttributeList getUserRecordData();
   public AttributeList getGeneralUserPreferences();
   public resultCode activateUserProfile(UserProfileName userProfileName);
   public resultCode deactivateUserProfile();
}
```

7.5.3 Delivery Context Handler Interfaces

7.5.3.1 Internal Interface

```
package DeliveryContextHandler;
public interface Internal {
     public DeliveryContext getDeliveryContext(String deliveryContextIdentifier);
}
```

7.5.3.2 External Interface

```
package DeliveryContextHandler;

import Personalization.Attribute;

public interface External extends Internal {
   public String[] listUserProfiles(String deliveryContextIdentifier);
   public Boolean storeServicePreferences(String deliveryContextIdentifier, Attribute[] attributeList, String userProfileName);
   public Boolean activateUserProfile(String deliveryContextIdentifier, String userProfileName);
   public Boolean deactivateUserProfile(String deliveryContextIdentifier);
}
```

7.6 Detailed Description of Relevant Technologies

7.6.1 XForms Controls supported by GUIML

The following sections describe all XForms controls supported by GUIML individually. A more complete explanation is given in [W3CXFORMS]. For simplification purposes, the examples use simple text labels. Using the complete GUIML specification, SMIL elements should be used in order to support multimedia data. However, for common input forms, such simple text input fields are sufficient.

7.6.1.1 The XForms Control Input

This form control enables free-form data entry. The input can comprise individual characters, numbers, or several words. Example:

```
<xfm:input model=”xforms” ref=”input”>
   <xfm:label>Name</xfm:label>
</xfm:input>
```

7.6.1.2 The XForms Control Secret

This form control is used for entering information that is considered sensitive, and thus is not to be displayed (echoed) in a visual display or played back in an audio interface. It is used to input a password or a PIN. Example:

```
<xfm:secret model=”xforms” ref=”secret”>
   <xfm:label>Password</xfm:label>
</xfm:secret>
```

7.6.1.3 The XForms Control Textarea

This form control enables free-form data entry, i.e. multiple lines of text. This form allows to input a complete text message, e.g. the body of an e-mail message. Example:

```
<xfm:textarea model=”xforms” ref=”textarea”>
   <xfm:label>Message</xfm:label>
</xfm:textarea>
```

7.6.1.4 The XForms Control Trigger

This form control is similar to the HTML element of the same name and allows for user-triggered actions. It is used to insert submit links into the document and to trigger the load action. The according link will be requested, when the user trigger this form, i.e. by pressing the label button. Example:

```
<xfm:trigger model=”xforms”>
   <xfm:label>Start</xfm:label>
   <xfm:action>
      <xfm:load xlink:href=”start.xml”/>
   </xfm:action>
</xfm:trigger>
```

7.6.1.5 The XForms Control Submit

This form control initiates the submission of all or part of the instance data belonging to it. The form instance data will be transmitted to the service according to the action, specified in the form declaration. Example:

```
<xfm:submit model=”xforms”>
   <xfm:label>Submit</xfm:label>
</xfm:submit>
```

7.6.1.6 The XForms Control Select1

This form control allows the user to make a single selection from multiple choices. The selection control contains a caption element and a list of items. Each item has a caption and a value that will be copied to the instance variable when the item is selected. Example:

```
<xfm:select1 model=”xforms” ref=”one”>
   <xfm:label>Select a credit card</xfm:label>
   <xfm:item>
      <xfm:label>VISA</xfm:label>
      <xfm:value>1</xfm:value>
   </xfm:item>
   <xfm:item>
      <xfm:label>Mastercard</xfm:label>
      <xfm:value>2</xfm:value>
   </xfm:item>
   <xfm:item>
```

```
      <xfm:label>American Express</xfm:label>
      <xfm:value>3</xfm:value>
   </xfm:item>
</xfm:select1>
```

Because, an item can contain also an action, the ‘select1’ control can be used to realize menus, i.e. a list of links. The action can also submit a form through send as shown in following example:

```
<xfm:select1 model=”xforms” ref=”multiple”>
   <xfm:label>Menu</xfm:label>
   <xfm:item>
      <xfm:label>Start</xfm:label>
      <xfm:action>
         <xfm:load xlink:href=”start.xml”/>
      </xfm:action>
   </xfm:item>
   <xfm:item>
      <xfm:label>End</xfm:label>
      <xfm:action>
         <xfm:load xlink:href=”end.xml”/>
      </xfm:action>
   </xfm:item>
</xfm:select1>
```

7.6.1.7 The XForms Control Select

This form control allows the user to make multiple selections from a set of choices. The instance variable will be set to all values, the user selected, separated by spaces. It is not allowed to use the space character in the value of any item. Example:

```
<xfm:select model=”xforms” ref=”multiple”>
   <xfm:label>Select you favourite fruits</xfm:label>
   <xfm:item>
      <xfm:label>Banana</xfm:label>
      <xfm:value>1</xfm:value>
   </xfm:item>
   <xfm:item>
      <xfm:label>Pineapple</xfm:label>
      <xfm:value>2</xfm:value>
   </xfm:item>
   <xfm:item>
      <xfm:label>Cherry</xfm:label>
      <xfm:value>3</xfm:value>
   </xfm:item>
</xfm:select>
```

7.6.2 GUIML Frames Configuration Set Example

An exemplary WWW layout configuration set, which transforms frames into a table-based representation, can look as follows:

```
<?xml version=”1.0”?>
<config>
   <css href=”…” />
   <frames>
      <table>
         <tr>
            <td>
               <frame id=”menu” />
            </td>
            <td>
               <frame id=”submenu” />
            </td>
            ...
         </tr>
         <tr>
```

```
            <td>
               <frame id=”main” />
            </td>
            ...
         </tr>
         ...
      </table>
   </frames>
</config>
```

The layout configuration consists of a surrounding ‘config’ tag. By using a ‘css’ tag with a ‘href’ attribute, an external style sheet document can be referenced to. The style sheet specifies how the content is to be displayed on the terminal. To define the frames for the device, a surrounding table tag and arbitrary number of rows that contain arbitrary number of columns is used. The tags may contain the ‘table’, ‘tr’, and ‘td’ attributes known from HTML. The columns contain a frame tag with an ‘id’ attribute. The id in the GUIML documents is used to place the frame content into the column with the same frame ‘id’.

7.7 List of Figures

7.8 List of Tables