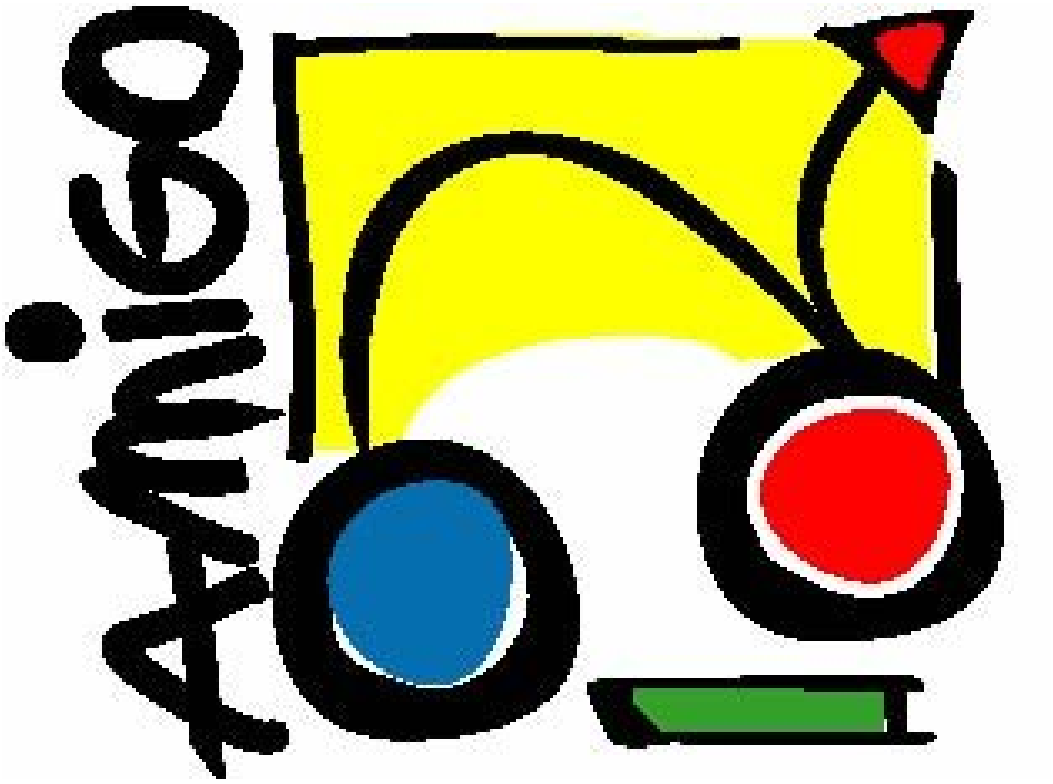


IST Amigo Project

Deliverable D4.1

Report on Specification and Description of Interfaces and Services

Public

Project Number	:	IST-004182
Project Title	:	Amigo
Deliverable Type	:	Report

Deliverable Number	:	Deliverable D4.1
Title of Deliverable	:	Report on Specification and Description of Interfaces and Services
Nature of Deliverable	:	Report
Internal Document Number	:	Amigo_D4.1_WP4_final.doc
Contractual Delivery Date	:	31 August 2005
Actual Delivery Date	:	3 November 2005
Contributing WPs	:	WP4
Author(s)	:	Maddy D. Janse (ed.), Peter Vink, Leo Rozendaal (Philips), Norbert Streitz, Carsten Magerkurth, Thorsten Prante, Carsten Roecker (FhG IPSI), Gilles Privat, Fano Ramparany (France Telecom), Henk Eertink, Tom Broens (TELIN), Juha Kela, Julia Kantorovitch, Elena Vildjiounaite (VTT), Christophe Cerisara (INRIA), Matthieu Quignard (INRIA), Mirko Barone, Sergio Di Marca (Italdesign – Giugiaro), Otilia Kocsis , Basilis Kladis (KNOW), Christian Ressel (IMS)

Abstract

This report presents the global design of the Intelligent User Services (IUS) that will be developed in the Amigo project. These services are Context Management Service, User Modeling and Profiling Service, Awareness and Notification Service, and User Interface Service. For each Intelligent User Service, the global concepts, modules and interfaces are explained, together with some examples of their dynamic behavior. The design of the Intelligent User Services starts from the user and technical requirements that were developed in the first phase of the Amigo project. A first outline is given of the privacy and personal security issues that are deeply embedded in the Amigo services, i.e., on how the users input and behavior can be translated into a model for the Intelligent User Services in which authentication and authorization service ensures protection.

This document follows on the architecture outline from Amigo Deliverable D2.3 by presenting the global design of the Intelligent User Services; it will be followed by a detailed design in Amigo Deliverable D4.2.

Keyword list

Connected home, ambient intelligence, service orientation, context awareness, user modeling and profiling, personalization, awareness and notification, rule based, user interface, multimodal dialogues, home information and entertainment applications, home care and safety applications, extended home environment, trust, privacy, awareness systems, user profiling, user modeling

Table of Contents

1 Introduction

This document presents the specification and description of components and interfaces of the Intelligent User Services (IUS) that are being worked on in the Amigo project. These services are: Context Management, User Modeling and Profiling, Awareness and Notification, and User Interface Service. They will be integrated as services into the Amigo Base Middleware. The Intelligent User Services facilitate the construction of composite services and applications that cover user requirements with regard to intuitive, personalized and unobtrusive interaction as envisioned within the ambient Amigo environment. Privacy and personal security aspects of such an environment are considered as a filter function that works for each Intelligent User Service. These filters provide rules for individuals in different contexts and for context changes. This implies that the handling of personal privacy and security aspects differs from the handling of the other Intelligent User Services in the Amigo system architecture.

This document follows from the work that was conducted in Amigo Work Packages WP1 and WP2. WP1 is responsible for acquiring the user requirements. In this work package, the initial scenario for the Amigo services and applications was evaluated with users to generate prioritized user requirements and to adjust the Amigo scenario to the user requirements (Amigo Deliverables D1.1 [1] and D1.2 [2]). WP2 is responsible for the system architecture. In this work package, the user requirements were used as input to derive technical requirements and to establish the initial architectural relationships between the Amigo Base Middleware and the Amigo Intelligent User Services (Amigo Deliverables D2.1 [3] and D2.3 [4]. Amigo deliverable D2.3 is an introduction to this document.

The Amigo Intelligent User Services are grounded within the user requirements that were acquired in WP1. There is a clear relation between the Intelligent User Services and the Amigo scenario. The Amigo architecture, developed in WP2 incorporates the Intelligent User Services such that, for example, their ontology descriptions can be used for composition. The Intelligent User Services use the Amigo Base Middleware that is being developed in WP3. For example, service definition, discovery, late binding, and interaction occurs through the mechanisms that are provided by WP3. WP3 also provides mechanisms for content distribution, and privacy and security, i.e., authentication, authorization and encryption methods. The Amigo demonstrators that will be developed in working packages WP5, Home Care and Safety, WP6, Home Information and Entertainment, and WP7, Extended Home Environment, exploit the Intelligent User Services.

First, in this introduction chapter, a short overview will be given of the results from WP1 and WP2 that form the base for the further elaboration and specification of the Intelligent User Services (IUS). This overview encompasses results from the state of the art analyses, the user studies, and the system architecture explorations.

Each of the following chapters will be dedicated to one of the Intelligent User Services. These chapters share the same structure. That is, a description of the service, including concrete examples derived from Amigo scenario elements, followed by a description of the architecture components needed for the service. The description of the service follows from the work in WP1 and WP2. The description of the components of the architecture, i.e., functionality and interfaces with other components, follows from the work in WP2 and relates to the work in WP3 that is being conducted in parallel.

1.1 User Requirements

The user requirements for the Amigo services and applications were derived from field studies, the evaluation of the Amigo scenario with users, and the analysis of the scenarios that are being used in relevant other projects. The list of user requirements for the Intelligent User Services was composed from the results of these studies.

1.1.1 Recommendations from State of the Art Research

Scenarios from relevant research projects in and outside Europe were analyzed (Amigo Deliverable D1.2 [2]). This analysis covered all application domains and Intelligent User Services for the Amigo project. Most of the scenarios in these analyses addressed only one implementation and didn't consider alternative solutions. The focus of these scenarios was on the user experience. Topics like security and context collection were under-represented. The recommendations generated from this analysis for the Amigo Intelligent User Services are:

- To create user interfaces that take the specific suitability of available devices into account. This requires a component based software platform that allows upgrading, adding, and replacing of software components. Furthermore, in order to automatically connect devices in the home to each other, private, public, and shared owner states, and transitions between these states have to be modeled. This also includes the modeling of cardinalities.
- To facilitate a natural user experience (e.g., Stre05 [5]), that is not bent to the requirements of a computer system and addresses the situation of several users that are concurrently interacting with multimodal systems.
- To provide users with parallel access to legacy interfaces, where applicable. That is, don't entirely replace users' prior knowledge, skills, and habits.
- To provide reliable communication channels that take privacy issues into account and that make best use of the available quality of service to enable social awareness and sharing of experiences. Provide strong and flexible privacy modes that protect intimate data, but also allow the easy sharing of this intimate information.
- To create an optimal intelligent room infrastructure that resides in the background and is only noticed by the users when they intend to interact with it. This requires interpretation and aggregation of context data from various sources to provide services through implicit user interaction. Furthermore, past behavior and preferences need to be taken into account in order to provide well-suited profiles.

1.1.2 Amigo Scenario Evaluation

The quantitative and qualitative evaluation of the Amigo scenario in WP1 provided prioritized user requirements (Amigo Deliverable D1.2 [2]). These user requirements were summarized in six categories. These categories are, in order of importance for users:

1. To maintain control and responsibility for organizing and maintaining the physical household
2. To cope with the overload of information and reduce the burden to search
3. To reduce the load of housekeeping chores
4. To assist with organizing the personal environment at home and between home and work
5. To assist with organizing the overall home environment
6. To support social relations, i.e., caring and staying in touch.

Confirming evidence from these extensive evaluations was obtained with regard to general user requirements including the very obvious, like the system should:

- Be easy to use and to configure – no need for programming by the user
- Not be used for surveillance
- Enable individual settings and preferences
- Be configurable by the user or service provider

- Be movable, in case of moving house
- Be extensible - easy to upgrade
- Be flexible
- Enable turning off individual features
- Be modular
- Be maintenance free (i.e., no need for maintenance by the user)

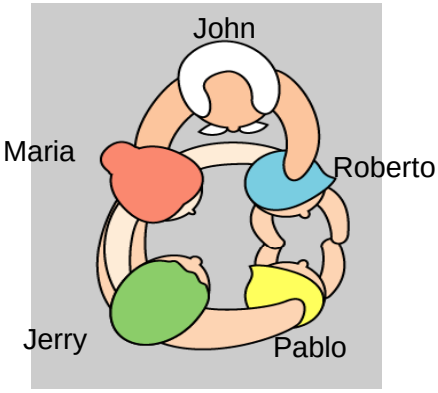
The full list of user requirements is given in Appendix A.

1.2 Refined Amigo Scenario

Based on the user requirements and the verbal feedback of the participants in the user studies, the Amigo scenario was rewritten. That is, the initial topics were refined towards more perceived user benefits, while the underlying technical challenges and application possibilities remained the same. Refinements concerned, amongst others, the problems with regard to privacy invasion that was mentioned by users. Also, the 'elderly role' is portrayed as active, instead of destitute and lonely.

The Amigo refined scenario portrays a day in the life of a family who has an operational ambient intelligence system. The scenario is composed of three scenes. These scenes correspond to the three application domains in the Amigo project, i.e., Home Information and Entertainment, Home Care and Safety, and Extended Home Environment. The scenario scenes are composed of elements. These elements are numbered from 1 – 15 in the scenario. The elements are used to describe and develop the Intelligent User Services.

Setting

Maria moved recently to Eindhoven with her husband Jerry and their two sons Roberto and Pablo. Before moving to Eindhoven, Maria and her family were living with her father, John, in Brussels. John is living alone. Maria and her father have installed Amigo systems in their houses to maintain close contact and still feel like being part of each other's daily family life. Their Amigo systems also help them with the daily housekeeping chores, their social agendas and taking care of their news and entertainment needs.	

Scene 1: Home Information and Entertainment

1. Maria's Amigo system works throughout the day for her. When she wakes up in the morning, Amigo starts her day with playing music from her preferred play list, her favorite radio host, showing her personalized news or general summaries of hot news topics. Amigo can order the program such, that it follows her everywhere in the house.
2. When Maria leaves the house, she can continue listening, as Amigo has downloaded the content on her personal device.

3. Amigo does the same for Jerry. Maria and Jerry can each have personalized news and entertainment programs at the same time independent of their location in the house. If Amigo is following them and if they happen to be in the same room, or with other members of the household, Amigo discretely stops and waits for new orders.
4. On occasion, Maria even starts a kind of karaoke session with her Amigo; they sing along with each other.
5. Amigo is Roberto's playmate for video games. Amigo keeps track of Roberto's play list, the status of his scores, and the people he has been playing with, downloads the games, and protects him from inappropriate games and content after consulting this with one of his parents.
6. When light and sound effects are important for Roberto's video games, Amigo adapts the home environment to the game by presenting lights, sounds and video throughout the room. Roberto is then in the center of the video game and can play by moving his body. The video is presented on a large wall, which also can show the people with whom he is playing or that are on-line and want to participate.
7. When Roberto's friends are coming over to his house for video gaming, Amigo downloads their profiles and integrates their game devices.
8. When Maria and Jerry are watching a film, Amigo adapts the home environment and creates an all-surround sound, light and video ambiance.

Scene 2: Home Care and Safety

9. Amigo functions as an usher-service for Maria and Jerry. This service recognizes people at the front and patio doors of the house and opens the door(s) for them if Maria and Jerry have authorized them. Amigo, like a real professional usher can notify the inhabitants, can take visiting cards and show personalized marks of the people that are present in the house, for example, a picture, a note, light or a color code.
10. Amigo can take over housekeeping tasks, starting and stopping the working of appliances at a desired time with the correct settings for duration, dosing, and temperature. It can even detect the presence of inappropriate objects in the appliances.
11. Amigo as kitchen chef downloads recipes and cooking programs to the kitchen and displays them for easy food preparation, i.e., cooking along with the video. Moreover, the recipes always take the status of the provisions in the kitchen into account.
12. Amigo maintains the overview of the food and household stock and generates shopping lists at predetermined time intervals. The shopping lists are personalized, but they take items that are on special offer, seasonal variations and nutritional balance into account.

Scene 3: Extended Home Environment

13. Roberto and his grandfather John have continued their habit of playing games together, watching a bit of TV and having their man-to-man chat. Amigo takes care of setting up the right ambiance. John's Amigo system is a modest version, with which he can be a participant in Roberto's games, just like Roberto's peers. Amigo selects the games that both John and Roberto like. They can look at each other and see what game moves are being made. Amigo can also set-up a video-conference for them in which they can watch TV together, show the newest acquisitions of their collections, or just tell their stories. With Pablo, the little one, John plays 'hide-and-seek' via this video communication.
14. Maria and her father have continued their habit of exercising together. Amigo sets up their exercise bikes, maintains their training schedules and lets them cycle through the video

landscape displayed on the video wall in Maria's home (the one used by Roberto for games). John watches the same scenery on his display, they see each other working hard, but Amigo makes sure that they each have their personalized exercise program.

15. Jerry likes to chat with Maria and his father-in-law while he is commuting from work and Maria and John have their work out. Amigo integrates his mobile phone or computer system with the video-conferencing system. He doesn't want to watch their work out, but occasionally he likes to draw Roberto in the audio exchange or discuss the recipes that Maria has downloaded in the kitchen.

1.3 Intelligent User Services

Intelligent User Services (IUS) are the building blocks for the realization of the application prototypes in the three application domains of the Amigo project, i.e., Home Care and Safety, Home Information and Entertainment, and Extended Home Environment. The position and relationships between the Intelligent User Services and the middleware and application layers in the overall Amigo architecture were adapted based on the results of WP1 and WP2. Figure 1.1 shows this overall view. Four abstract Intelligent User Services are defined in the Amigo middleware architecture. They are detailed in this document. These services are:

- Context Management Service: This service collects, uses and predicts the context information. The goal is to provide the right information to enable the system to adapt as good as possible to the conditions of the physical context and the behavior of the persons and devices in it.
- User Modeling and Profiling Service: This service extracts user preferences from user actions or utterances. The goal is to adapt the system to these user preferences.
- Awareness and Notification Service: This service provides a mechanism that exploits information provided by different sources and presents this information in an appropriate fashion to different stakeholders; it also provides mechanisms that allow users to stay in touch with friends, and have a feeling of being connected with them.
- User Interface Service: This service handles the devices to present the contents, interaction modalities and explicit and implicit user interactions.
- Privacy and Security is orthogonal to these services. It works as a filter function for each Intelligent User Service and provides rules for handling people's privacy in different contexts.

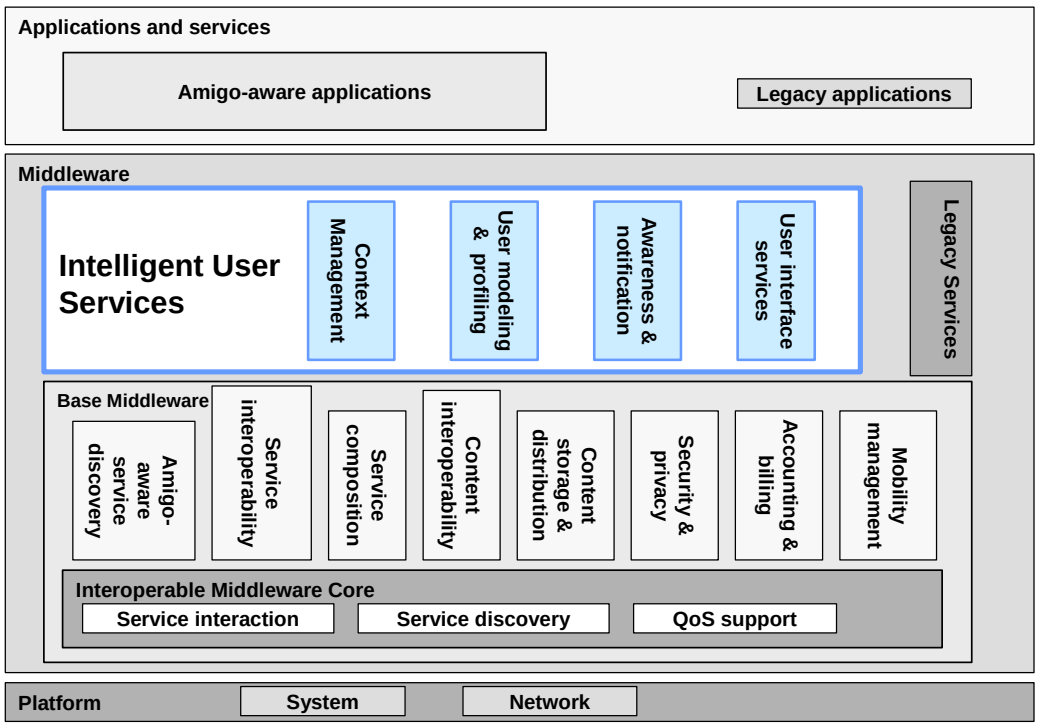


Figure 1.1: Amigo schematic architecture

In Amigo Deliverable D2.3 [4], two more services were identified:

- Content Providing Service: This service provides and accesses different types of content and tailors it depending on context information and user profiles. This service is now subsumed in the Amigo Base Middleware in the Content Storage and Distribution component. It will not be discussed in this document.
- The Privacy and Personal Security Service: This service ensures the personal privacy and security protection of the user. It was decided not to implement this functionality as a separate Intelligent User Service but as an orthogonal mechanism, affecting the implementation of the other Amigo services. This will be discussed in chapter 6 of this document.

1.4 Dependencies between Intelligent User Services

Figure 1.2 presents a very global overview of the Intelligent User Services and their main interfaces. The service-oriented nature of Amigo (see Amigo Deliverable D2.1 [3]) is depicted here in a bus-like architecture: Services can connect to the Amigo Service Framework and offer their services. Other services such as those described in Amigo Deliverable D3.1 [6] (Vol. C) are also connected; e.g., Authorization and Authentication Service. The Service Discovery Infrastructure described in D3.1 [6] (Vol. B) serves as framework. Composite services and applications use the services in a similar way. The Amigo Service framework has to function real-time and is assumed to be ‘always on”.

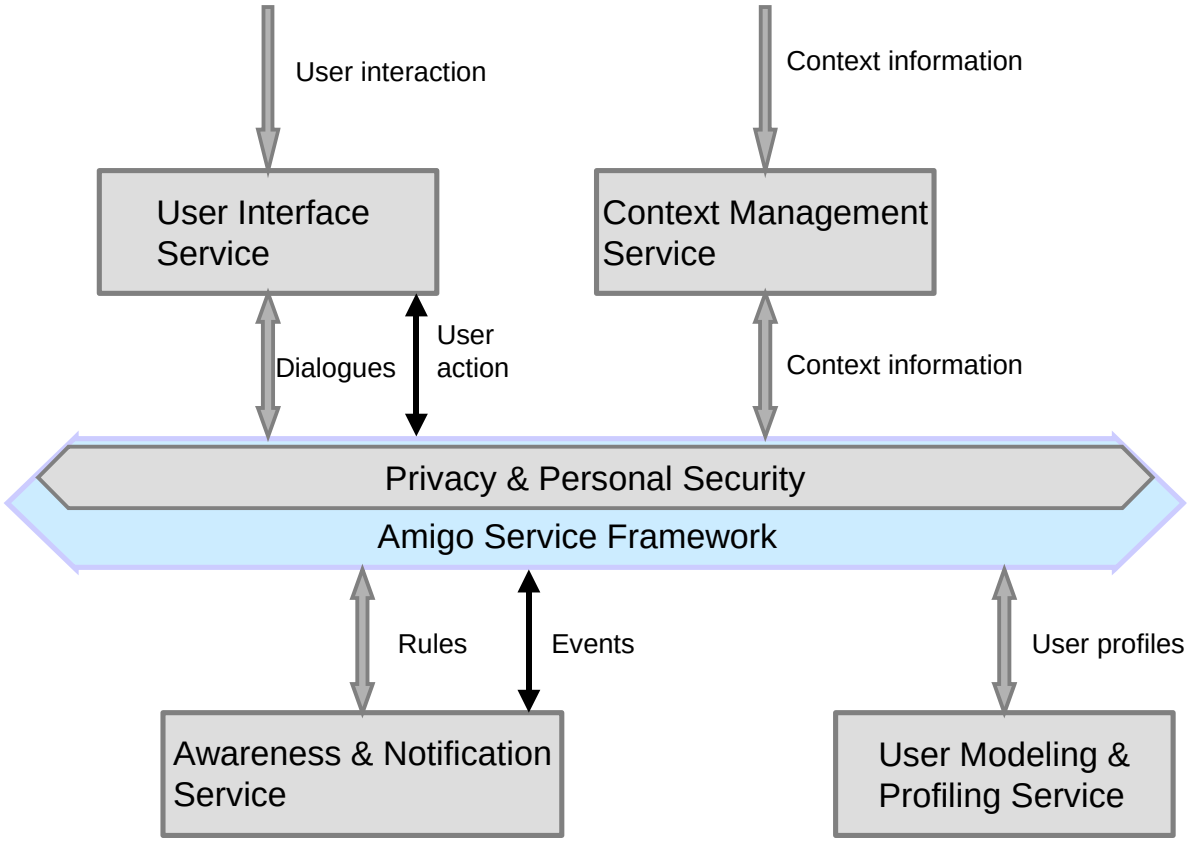


Figure 1.2: Amigo Service Framework

Each of the four Intelligent User Services models a specific attribute, i.e., dialogues, context information, rules and user profiles (depicted in the arrows), which is closely connected to the offered functionality.

- Context Management Service maintains and offers Context Information (e.g. sensor information).
- User Interface Service offers the modeling of User Dialogues (e.g. via device, speech, gestures, and combinations).
- Awareness and Notification Service maintains Notification Rules, which can result in events.
- User Modeling and Profiling and Service maintains a User Profile and offers the personal settings of users

It is possible for more services to co-exist. In this document however, services have been described in isolation. Furthermore, we abstract from the implementation platforms by assuming that these services will publish and use other services provided by the middleware (Amigo Deliverable D3.1 [6]).

The Privacy and Personal Security works as a filter function for each Intelligent User Service.

1.5 Relationship with the Amigo Base Middleware

The Amigo Base Middleware (as defined in Amigo Deliverable D3.1 [6]) provides a framework for the definition, discovery, and late binding of services. All Intelligent User Services will be defined, registered and discovered through the features of the Amigo Base Middleware. This means, for instance, that the context management functions will use the vocabulary defined in Amigo Deliverable D3.1 [6], and that the application can discover and invoke Intelligent User Services through the mechanisms defined in D3.1. Furthermore, all Intelligent User Services will use the Security and Privacy services defined in WP3, in order to guarantee features like single sign-on and integrated authorization.

In the detailed design document (Amigo Deliverable D4.2 [7], forthcoming), it will be indicated which functional elements will be 'Amigo services', i.e., discoverable through the WP3 framework, and which elements are only used internally within an Intelligent User Service and therefore need not be managed by the Amigo Base Middleware.

In the remainder of this document, the Amigo Service Framework is assumed, though it is not mentioned explicitly. In other words, the external interfaces that are offered by one of the Intelligent User Services, are presented via the Amigo Service Framework and each Intelligent User Service has exported functionality for its discovery, registration, etc.

1.6 Document Structure and Conventions

This document describes the main components within the services. It describes what these components do and not how they reach their result. Where possible, UML static component/package diagrams will be used. The interfaces to other services, applications and to internal components are described at top level. For this, UML static structures and sequence diagrams will be used. The detailed design will be worked out in Amigo Deliverable D4.2 [7] (forthcoming). Amigo Deliverable D4.2 will show state and activity diagrams for components that define how particular components reach their results, define what the internal interfaces are, show UML deployment scenarios to illustrate replication and distribution aspects, and define which components are Amigo services that can be queried, initiated and managed by the Amigo Base Middleware.

1.6.1 Conventions

Components and Interfaces (UML Component Diagram)

In this document, a *Component* is not a «software component», as defined in a regular UML component diagram. It shall be understood as a "module", or more precisely, as a "unit of composition with contractually specified interfaces and explicit context dependencies only" (Szy98 [8]).

The main idea is to *combine* components, which might pre-exist, in order to build the Amigo system. As a guideline, there shall not be many more interfaces than the number of components. This will be shown graphically by using class notation, with a stereotype <<component>>. A classical cylinder object depicts storage components.

A *depends on* B means that B provides some functionality, which is used by A. Graphically this will be shown with a dependency arrow. Stereotypes could be, for example <<uses>> or <<notifies>>.

B provides an interface I means that B provides some functionality, which can be accessed by any module via a few, simple "functions" of I. The goal of the interface is to hide all the internal complexity of B (keyword = simplicity), so that any module can access these functions without caring at all about how B works (keyword = decoupling).

Interfaces and *Components* are also described as a UML static structure in order to detail the functionality.

For an overview of the symbols that are used, refer to Figure 1.3: Notation used within Amigo component diagram.

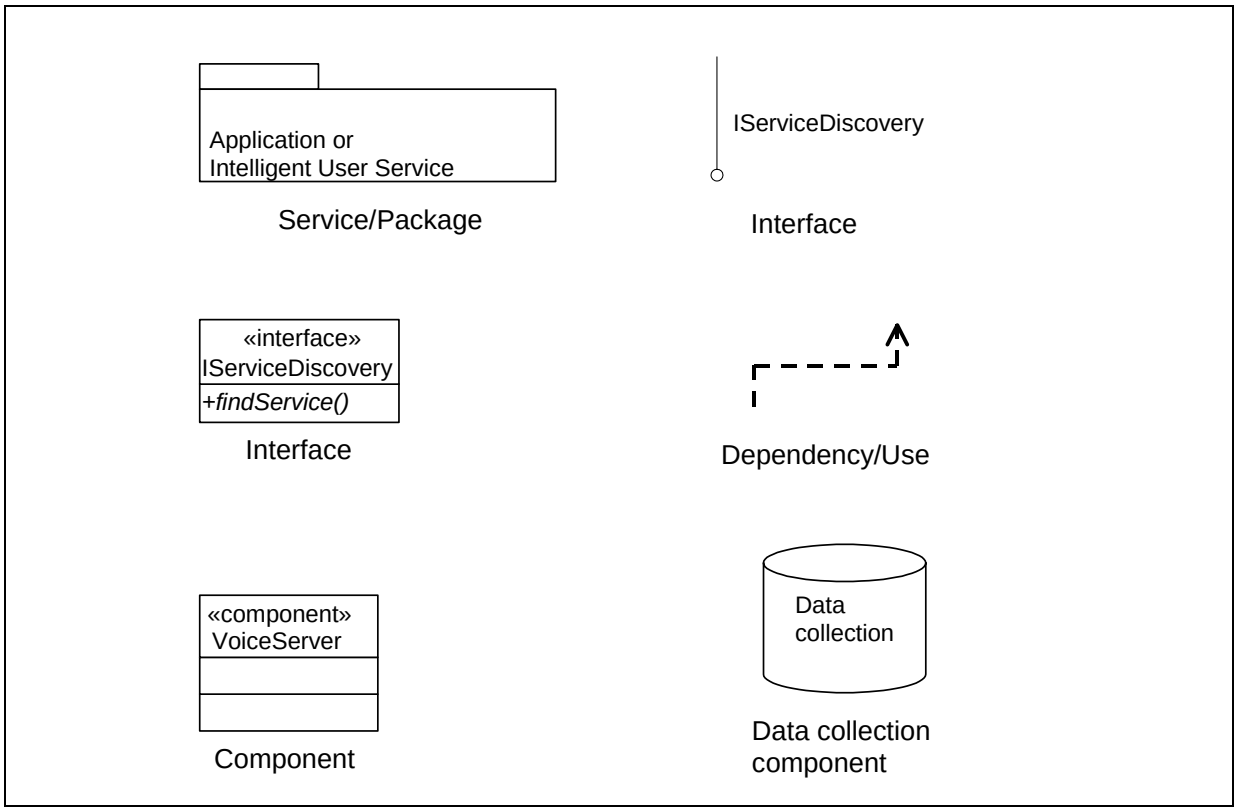


Figure 1.3: Notation used within Amigo component diagram

Dynamic Behavior (UML Sequence Diagrams)

To show interactions in the context of a scenario, with other services and within the service UML sequence diagrams are used.

For an overview of the symbols that are used, refer to Figure 1.4: Notation used within Amigo sequence diagram.

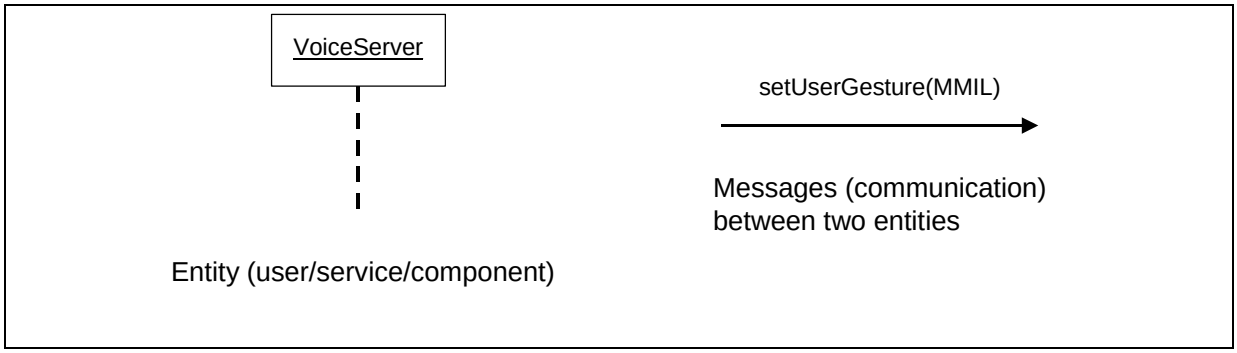


Figure 1.4: Notation used within Amigo sequence diagram

For functionalities that require being synchronous, we use names starting with “set” or “get”, while for asynchronous functionalities the names start with “request” or “update”. The following conventions for naming are used:

- InitialCapitals for components
- All Interfaces start with I; e.g., IUserControl
- No capital for methods and attributes
- Nouns for static objects
- Verbs for methods
- References to these names in the text of the document are in italics
- References to methods or properties of components are given as: ComponentName::methodName.

In the next chapters the Intelligent User Services will be described. Their relation with regard to the Amigo scenario and user requirements will be given. For each Intelligent User Service, the service architecture is described by means of a conceptual data model, a design overview, the interfaces and dependencies will be outlined and examples of dynamic behavior will be used.

2 Context Management Service

2.1 Introduction

The role of the Context Management Service is to acquire information coming from various sources to combine these pieces of information into "context information", and to make this context information available to Amigo services. The context sources range from physical sensors, user activities, applications in process, to Internet applications. The Context Management Service enables the Amigo services to become context-aware, e.g., to support context-aware service discovery and context-aware service composition. The Context Management Service describes what users are currently doing, which devices and applications are in use and where they are located. It uses this knowledge and knowledge about the past to predict the changes in the context to enable appropriate adaptations. The Context Management Service collects, uses, and predicts the context information. This context information is usually dynamic, retrieved from appropriate sources, collected and organized when needed.

The Content Management Service has to provide the right information to enable the system to adapt as good as possible to the conditions of the physical context and the behavior of the persons and devices in it. As the amount of raw data generated, for example, by sensors and sensor networks may become overwhelming, it is necessary to aggregate, pre-process and interpret the context data before they are exploited.

One of the major goals of the functionality of the Context Management Service is to predict user needs or contexts for proactive offering of functionality to users. One major approach for making such meaningful predictions from context data is to use stored histories of user's interactions in context (May05 [9]).

The following general definition of context is used in the Amigo project:

> '*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, the device and application themselves*' (Amigo Deliverable D2.1 [3], p.57).

2.2 Context Management in the Amigo Scenario

The description of the Context Management Service is based on scenario elements. These elements provide the application topics that will be worked on in WP4, Intelligent User Services.

2.2.1 Amigo Scenario Examples

To develop the architectural dependencies and interfaces for the Context Management Service, several scenario topics have been identified as being particular relevant. These topics and their related scenario scenes and elements are (numbers refer to the scenario elements presented in Chapter 1.2):

1. Usher

Scene 2, element 9

> Amigo functions as an usher-service for Maria and Jerry. This service recognizes people at the front and patio doors of the house and opens the door(s) for them if Maria and Jerry have authorized them. Amigo, like a real doorman can notify the inhabitants, can take visiting cards and show personalized marks of the people that are present in the house, for example, a picture, a note, light or a color code.

2. Personalized Content Selecting

Scene 1, element 1

Maria's Amigo system works throughout the day for her. When she wakes up in the morning, Amigo starts her day with playing music from her preferred play list, her favorite radio host, showing her personalized news or general summaries of hot news topics. Amigo can order the program such, that it follows her everywhere in the house.

Scene 1, element 2

When Maria leaves the house, she can continue listening, as Amigo has downloaded the content on her personal device.

3. **Adaptive Content Provisioning**

Scene 1, element 3

Amigo does the same for Jerry. Maria and Jerry can each have personalized news and entertainment programs at the same time independent of their location in the house. If Amigo is following them and if they happen to be in the same room, or with other members of the household, Amigo discretely stops and waits for new orders.

4. Caring & Sharing

Scene 3, element 13

Roberto and his grandfather John have continued their habit of playing games together, watching a bit of TV and having their man-to-man chat. Amigo takes care of setting up the right ambiance. John's Amigo system is a modest version, with which he can be a participant in Roberto's games, just like Roberto's peers. Amigo selects the games that both John and Roberto like. They can look at each other and see what game moves are being made. Amigo can also set-up a video-conference for them in which they can watch TV together, show the newest acquisitions of their collections, or just tell their stories. With Pablo, the little one, John plays 'hide-and-seek' via this video communication.

Scene 3, element 14

Maria and her father have continued their habit of exercising together. Amigo sets up their exercise bikes, maintains their training schedules and lets them cycle through the video landscape displayed on the video wall in Maria's home (the one used by Roberto for games). John watches the same scenery on his display, they see each other working hard, but Amigo makes sure that they each have their personalized exercise program.

The major role of the Usher is to identify which persons are at home, where they are in the home and what the context of their behavior is. This requires acquisition and adaptation of information, i.e., when and where people move around in the house, and enter and leave the house.

The major role of the Personal Content Selecting is to identify favorite and personalized information (in this case music) and to present this on the most appropriate available device. This implies that the content presented has to be adapted to the change in context re. location, device, and social environment when the users move around. This extends to the user leaving the house as well as to other people entering the user's direct environment. The system has to predict this and to adapt the content presentation to, for example, a mobile device. The system needs to predict, for example, which content needs to be pre-loaded to the mobile device.

The notions of favorite songs, personalized news, summaries of news as well as what news to be downloaded to a mobile device can be interpreted with the help of an enhanced understanding of the users' interactions over time and embedded into and related to other interactions happening in parallel, before, or afterwards.

The major role of the Adaptive Content Provisioning is to identify the social context, i.e., other persons in the same room, know the individual and the joint preferences for content to predict what to present or establish other desired behaviors.

The major role of the Caring & Sharing is to identify the required ambiance from acquired and aggregated information from, for example, time of day, who is where, user profiles, allocation, detection and knowledge of devices in at least two different context environments, i.e., two homes, or a home with users and other users 'on the move'. The 'exercise-together' scene can be envisioned as two users who are each exercising in their own home, but also as one user exercising at home and the other user exercising outdoors. Furthermore, the systems at both sides don't need to be identical and the applications have to adapt themselves appropriately. This implies that information needs to be acquired and aggregated about the availability and capabilities of these devices and that predictions need to be made with regard to the best presentation. Prediction is also required to adapt to context changes at either side and to prevent intrusions.

2.2.2 Related User Requirements

The following most relevant user requirements from the list in Amigo Deliverable D2.1 [3], Chapter 9[1], (Appendix A in this document) are addressed in the topics for the Context Management Service:

16: The system should provide concurrently the appropriate information to the right persons and devices for the appropriate occasion at different locations.

26: The system should protect against abuse, intrusions, loss of data, and house hackers

27: The system should provide controllable access and respect individual preferences and authorities

29: The system should take context/environment conditions into account and be aware at any time of the local situation

32: The system should take implicit social rules into account

34: The system should enable communication with multiple people at the same time.

35/36/37: The system should support keeping in touch, feeling connected with family and friends, and support trusted relationships.

The user benefits identified in Amigo Deliverable D1.2 [2], that are relevant for these application topics are:

- Recognition of profiles of family and friends at the entrance doors. Hands and key free entrance and exit to the house. Personalized notification of presence and availability.
- Personalized content anywhere, anytime and adapted to social conditions.
- Caring & sharing – sharing activities and content independent of location with family and friends.

2.3 Context Management Service Architecture

2.3.1 Introduction

To enable the development of context-aware services or applications two requirements are crucial. These requirements are:

[1] The numbers refer to the numbers used for user requirements in D2.1, Chapter 9

- For the Amigo middleware architecture: the ability to use context information while looking up a service and
- For the Amigo application level services: the ability to exploit context information for its internal processing.

In WP2 (Amigo Deliverable D2.3 [4]) an abstract overview of the Context Management Service in relation to the other Intelligent User Services, the Base Middleware and the application layer was given. Context management and context prediction are the crucial components of this service. They address specific requirements with regard to handling real-time events, maintaining persistence by storing and retrieving data.

The main functions of the Context Management Service is to acquire rough data from sensors and other context information sources (UI interfaces, services, external or legacy context information sources), to aggregate and abstract these data into relevant context information, and make this information available to the other Amigo Intelligent User Services and applications. Storing and exploiting context histories (Pra05 [10] and activity patterns make it possible to predict future contexts.

A major requirement is the ability to use the Context Management Service as a mediator between entities, which provide context information (e.g. sensors) and entities, which consume context information (e.g. services). It is an Amigo requirement that context information must be obtained from arbitrary Amigo devices in the user's environment. Since, a centralized Context Management Service that is able to function as a broker between context information consumers and context information providers may not always be available, the context management architecture should also support a de-centralized Context Management Service, i.e., peer-to-peer context management functionality. Figure 2.1 shows the centralized and distributed peer-to-peer approach for context provisioning. The arrows in Figure 2.1 show the context information abstraction order. In the peer-to-peer configuration, context information is managed in a distributed way, which means that context information is shared among entities. More specifically, low-level context information is taken care of by context information providers such as sensor devices and high-level context information is taken care of by context information consumers such as application services.

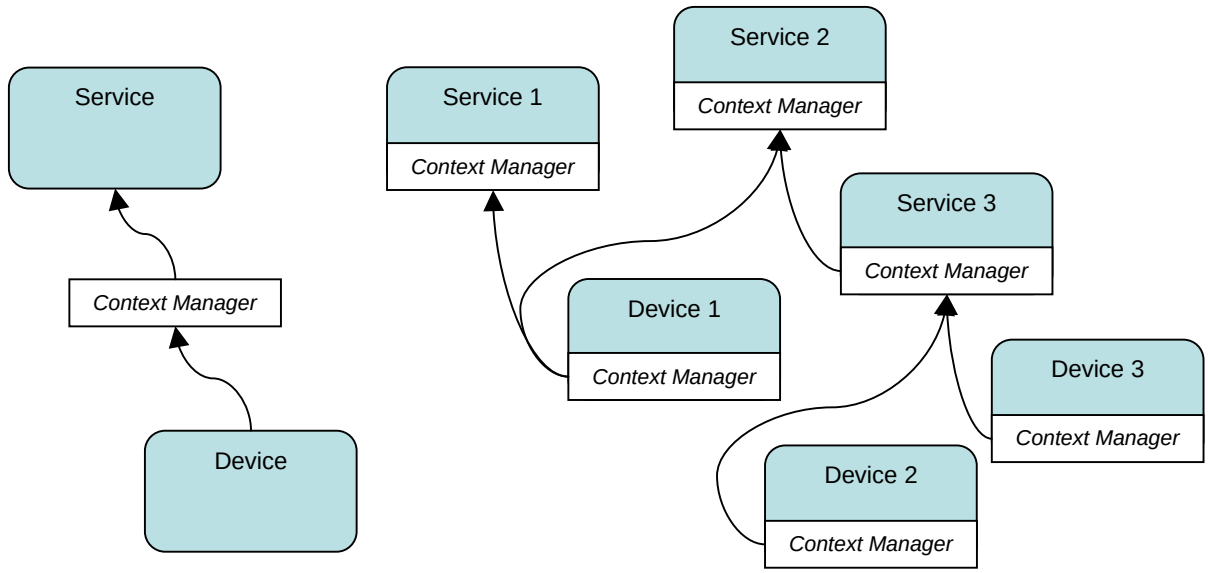


Figure 2.1: Centralized and peer-to-peer context provisioning

2.3.2 Conceptual Data Model

Intelligent User Services operate by handling the minimum amount of information that is necessary for their execution and for their adaptation to their changing environment. Intelligent User Services reason on their own representation of context that they have to maintain to remain up to date. Hence, they have to exchange and enrich their own set of information. They have to inform their model and fetch other context information through the Context Management Service by making queries. Those queries are expressed according to the context model of the requester and have to be satisfied according to the same model. In some cases, the context model and the context information are provided by another service or sensor, they may have the same name, but might have a different semantic meaning.

Context Information is based on multiple data types that are defined in the Context Ontology as elicited in Amigo Deliverable D3.1a [6]. Actual context information is made of instances of those types and assertions about how those instances relate to each other.

The minimal set of slots to be used for representing a piece of context information includes the following items:

- Name
- Type (defined in the Context Ontology)
- Value
- Timestamp
- Subject (context of what: e.g. a room, a device, a person)
- Origin (the entity that created this context information value)
- Confidence (notably for derived information: how sure are you that it is correct?)

This representation will be flexible enough to support various data such as raw sensor data, or grouped sensor data (e.g., time series data reduced to start and end time) and complex situations (e.g., Maria watching a movie, Jerry cooking, or having a family breakfast).

2.3.3 Design Overview

Figure 2.2 depicts the abstract overview of the Context Management Service architecture.

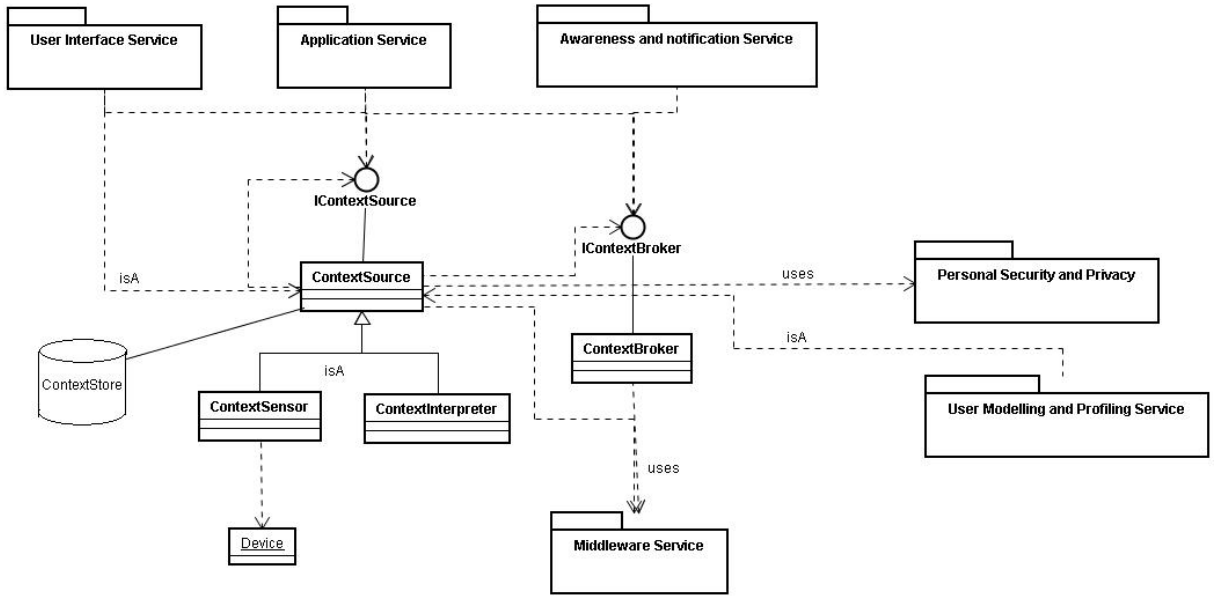


Figure 2.2: Context Management Service architecture

The Context Management Service architecture consists of the following components (Figure 2.2):

- *ContextSource*
- *ContextSensor*
- *ContextInterpreter*
- *ContextBroker*

2.3.3.1 ContextSource

ContextSource is the functional module that provides access to local context information (through the *IContextSource* interface). It can represent sensors (*ContextSensor*) or more elaborated context aggregation services (*ContextInterpreter*).

The *ContextSource* is the single point of access for context retrieval. It returns context sources, which on their turn are either representations of real-world context providers, e.g., sensors, or context interpreters, which infer higher level context information from sensor data.

The *ContextSource* must have the capability to store and access context information in a consistent, yet flexible manner. In many cases, especially when context data are received from heterogeneous sensors, it is desirable to maintain persistency for the data. A persistence mechanism is any technology that can be used to permanently store objects for later update, retrieval, and/or deletion. Possible persistence mechanisms include flat files, relational databases, object-relational databases, hierarchical databases, network databases, and object databases. The *ContextStore* in Figure 2.2 uses such a mechanism.

The main purpose of a local persistence module is to provide a mechanism that makes context data available, even if the context information that is received from sensors and integrated is transient.

2.3.3.2 ContextSensor

ContextSensor abstracts the functionalities of physical sensors, so that the hardware and transport level details of the sensors remain transparent to the Context Management Service and, even more important, to the clients of this service. For example, if a light detector transmits its results to its host computer using a serial line connection, the *ContextSensor* component will act as a gateway that provides a standard communication interface to the light sensing service.

Contextual data can be obtained from several sources: simple sensors, complex sensors, or other devices, such as, consumer electronics devices or domotic appliances. That is, this context information can be acquired from multiple, distributed and heterogeneous sources.

2.3.3.3 ContextInterpreter

Intelligent User Services have to update their context model with information from the Context Management Service that has the same semantics as the information in their own context model to maintain consistency in their context model, reasoning and belief. The *ContextInterpreter* will compare and find these relationships. These operations are conducted with a semantic modeling language and the use of alignment techniques. Thanks to these alignment techniques, the concept that is defined in a context model that corresponds to a concept defined in another context model can be retrieved.

2.3.3.4 ContextBroker

The *ContextBroker* module is responsible for finding context providers of particular types. It implements the *IContextBroker* interface.

2.3.4 Interfaces

The Context Management Service Architecture consists of the following interfaces:

- *IContextBroker* Interface
- *IContextSource* Interface

2.3.4.1 IContextBroker Interface

The *IContextBroker* interface provides access to context information for user applications, Amigo services or other third party services. It provides access control to context sources, is able to lookup the proper context source for a particular service, and is therefore also responsible for managing a collection of context sources.

Every Amigo device may provide context information. The main functionality that is provided by the *IContextBroker* interface is:

- Discovery of locally available context sources
- Registration of context sources
- De-registration of context sources.

This is reflected in the following diagram (Figure 2.3) of the *IContextBroker* interface.

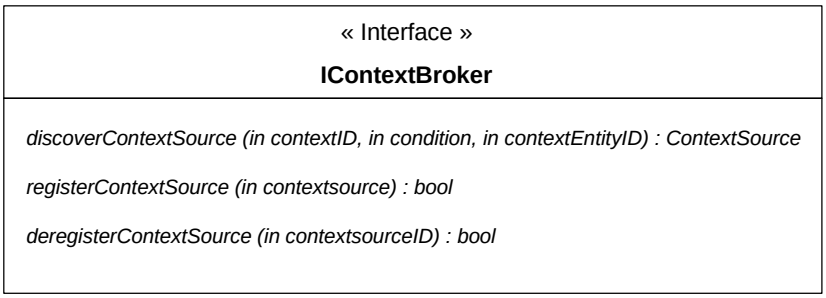


Figure 2.3: IContextBroker Interface

It is assumed that the *IContextBroker* interface of a particular device is discoverable through the Amigo service discovery mechanisms. We abstracted in this definition security aspects like authentication and access control, which should be handled by the Context Manager Service.

The *discoverContextSource* method is used to find certain contextual information providers. This method takes a *contextID* (e.g., 'location'), a *condition* (e.g., 'accuracy < 0.5m') and a *contextEntityID* (e.g., 'John'). It will query its local context repository for available context sources. When an appropriate *ContextSource* is found it will be returned. The *contextID* and entities involved in the conditions should be concepts specified in the Context Ontology. The following is an example of context acquisition for the location of a person named John: *discoverContextSource* ("location", "accuracy <0.5m", "John"). This method invocation could return a RFID location context source while *discoverContextSource* ("location", "accuracy <10m", "John") could return a GPS location context source (less accurate then RFID).

The *registerContextSource* method provides a way to register new *ContextSources* to the registry of the *IContextSource*. After registration, the *ContextSources* are discoverable for other parties. The *deregisterContextSource* will delete the *ContextSource* from the registry.

2.3.4.2 *IContextSource* Interface

A *ContextSource* (see Figure 2.2: Context Management Service architecture) is an abstraction that provides access to context information in a synchronous or asynchronous way.

Examples of context sources are (typically wrapped) sensors, providers of aggregated information (e.g. collections of QoS parameters), or interpreted context information. For example, derive the room of a particular user; such a source typically uses reasoning based on knowledge databases that contain information about the structure of the building and other context sources for obtaining the current location of users.

The services provided by a *ContextSource* [2] are depicted below in Figure 2.4:

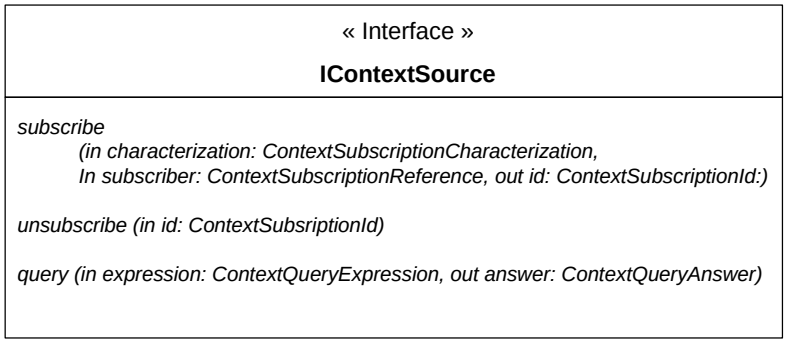


Figure 2.4: IContextSource Interface

The operation *subscribe* is used to register a notification request, the operation *unsubscribe* is used to withdraw a given notification subscription and the operation *query* is used to select specific context information instances. The definition of specification languages to define context subscription characterization, context query expression and context query answer is a topic of research in the Awareness project (AWA04 [11]). Potential clients of the *IContextSource* Interface are (i) specific applications and (ii) other *ContextSource*s that offer context information that is derived of this *ContextSource*. Note that a *ContextSource* can be registered at one or more context providers.

ContextSource exchanges information through well-defined data formats and protocols. In particular the asynchronous format should be compatible with the IETF presence model (as defined in RFC2778), and presence data formats (as defined in RFC3863). These models are currently being implemented in particular in presence systems like Jabber, and in the SIP/IMS messaging environments.

The *ContextQueryAnswer* contains this data, and also metadata that describe quality parameters (to take properties like timeliness or confidence into account).

2.3.5 Dependencies

The Context Management Service depends on the following middleware services:

- Security: Context information should be protected against attempts to its integrity
- Persistency management: Context information histories requires persistent storage management, mainly for history storage
- Event management: Context providers need to be able to create context change specific events and to notify interested parties on these changes
- Peer-to-peer support, service discovery: The context broker will use discovery mechanisms for registering and searching context providers

[2] This is strongly based on the Context Provider Service defined in the Awareness project (AWARENESS –D2.1, [11])

- Messaging: Communication between context consumers (application services, Intelligent User Services, or simply other context providers) and context providers will be based on existing middleware level communication services.

The Context Management Service depends on the following Intelligent User Services:

- Awareness and Notification Service for:
  - Presence information that is either stored explicitly, provided by context provider services, or that could be directly derived from context provider information.
  - Changes of this presence information. Such changes will trigger events to which the Context Manager Service should subscribe.
- Privacy and Personal Security: Access to context information is granted only to authorized clients.
- User Interface Service: Some user interface services provide context information, thus they might provide the context source interface.
- User Modeling and Profiling: User profile is the context of the user and the User Modeling and Profiling Service should implement the context source interface, (*IcontextSource*).

Some applications might be content providers; they export their main states, e.g., idle, running, or to be logged as context information.

2.3.6 Dynamic Behavior

We specify two dynamic behaviors using UML sequence diagrams. The first behavior involves the introduction of a new context provider (e.g. a new sensor) and the query of a context consumer addressed to this newcomer. The second behavior involves the introduction of a new context provider and the subscription of a context consumer to some context change notification. For example, a newly available provider of context information is introduced by switching on the TV, i.e., the context: current program. An application (e.g., the profile service of Maria that discovers that she is in the room and wants to know what she is watching) asks the tuner what is on. The TV is switched off and the context is not available any more (only its history). The context change events that are matching the interest of the context consumer are sent to the context consumer. Both behaviors terminate by deregistration of the *ContextSource* (e.g., the *ContextSource* consciously resumes its operation).

2.3.6.1 Context Query Behavior

The following diagram shows an example of the dynamic behavior of *ContextQuery*.

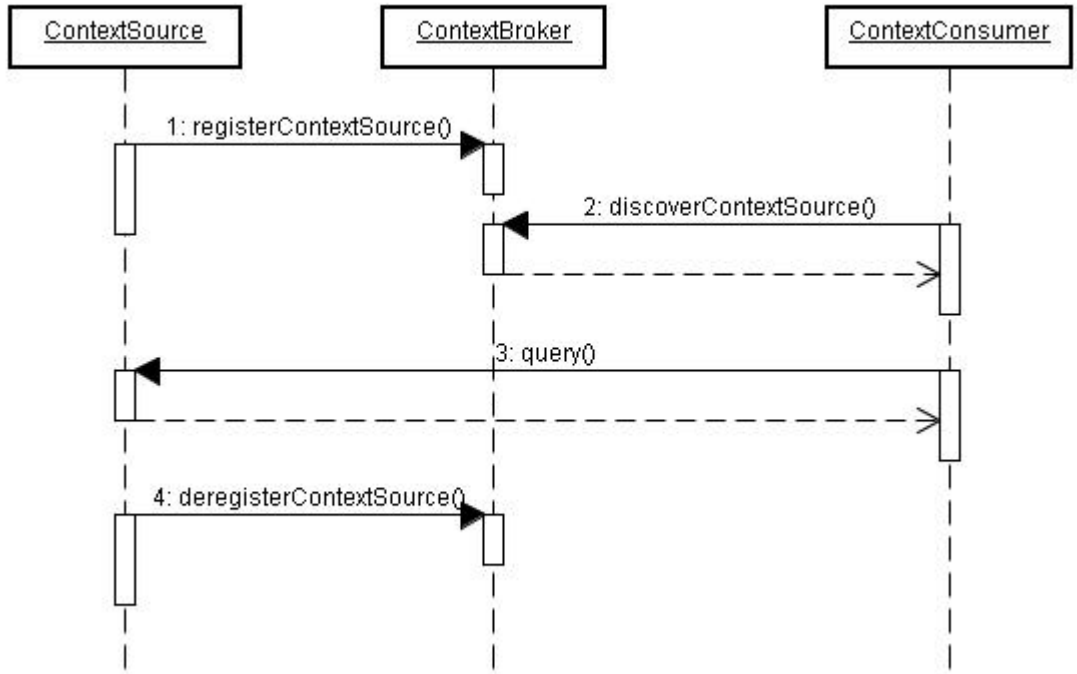


Figure 2.5: Example of Context Query Behavior

The *ContextConsumer* entity is a context aware application or service.

2.3.6.2 Context Change Notification Behavior

The following diagram shows an example of the dynamic behavior of Context change Notification.

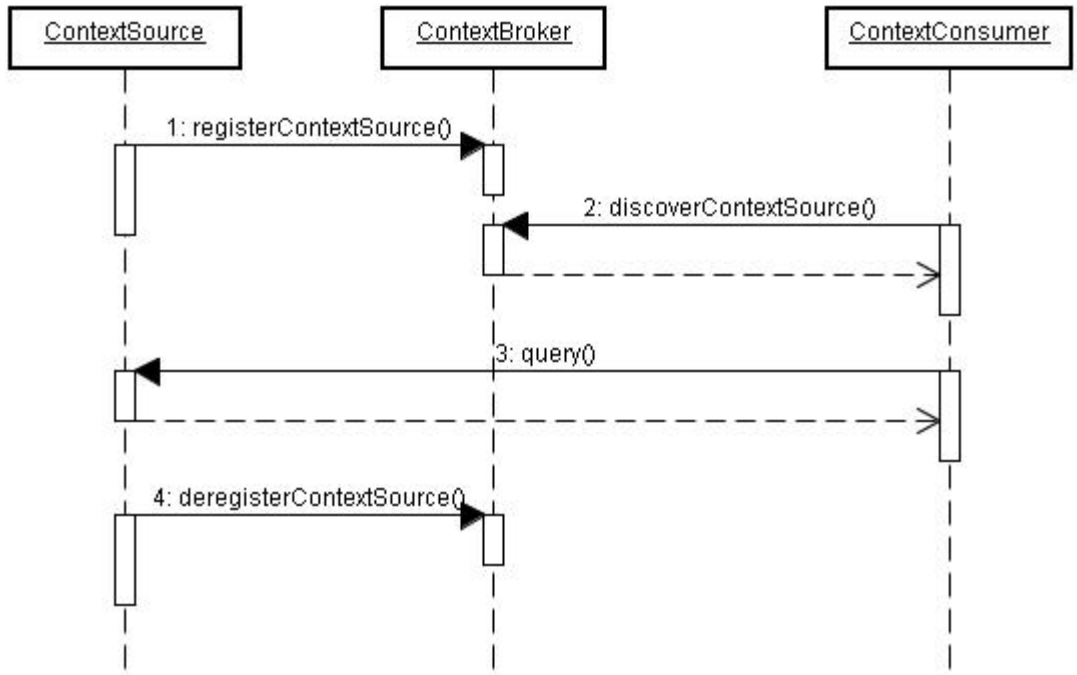


Figure 2.6: Example of Context Change Notification behavior.

3 User Modeling and Profiling Service

3.1 Introduction

The User Modeling and Profiling Service (UMPS) models and maintains user profiles by means of personalization of content and user interfaces, by combining this information in user profiles and by keeping histories. It handles multiple user profiles and more complex or aggregated information and publishes new information. This service enables the applications to become personalized.

The current trend in user modeling and profiling research has evolved from representation of groups of users using a certain system in certain conditions (GEM04 [12]), to personalization of these systems towards individual users preferences and requirements. Personalization applies to different system components and application domains, but in particular to user interfaces (e.g., graphical, gesture, voice-based) and content (e.g., multimedia). Personalization depends on the initial knowledge of the system about its potential users and the mechanism used to learn user's behavior and preferences (Rich98 [13]). But, it also depends heavily on the context in which the system is used. Therefore the system should be adaptable to the context. In addition, system behavior and adaptation to multi-user environments must be considered.

Personalization of user interfaces comprises several advantages. It can be used to adapt the appearance and structure to the user and the user's way of behaving to provide interfaces that are more effective and acceptable for all users. Today, adaptation of the appearance is commonly used in all kinds of graphical user interfaces. The user can choose, for example, background images, color of menu bars and dialog windows, and sort menu entries into customized submenus. Normally, this sorting has to be done by the user. But it is also imaginable that this sorting is done automatically and the user has just to describe the idea of different submenus (e.g. into this submenu put every multimedia device). This approach would support flexible systems with dynamical services like the Amigo system. Such flexibility in the user interface could also allow adaptation of the input recognition part of user-system interaction acts to help to prevent misinterpretations. For example, a speech interface could be adapted to the user's voice and expressions to reduce the error ratio of the recognition and understanding subsystems. Furthermore the information about user behavior can be used to predict future inputs and offer shortcuts to reach the interaction goal in a more effective way (e.g. less interaction steps).

Personalization of multimedia content implies presentation of different multimedia items (e.g., videos, music or news) to different users based on their preferences. For example, one person likes action movies, and another person does not like violence scenes at all; one person is interested in news about politics, and another one about football, and so on. However, these personal preferences depend also on the current situation (context) of the person. There could be multimedia content which parents do not want to watch in the presence of their children, but which they would prefer when they are alone; even the person who almost never watches news about political events most probably will want to see the news about such important events as the result of a presidential election or a major terrorist attack; generally, the presentation of the news should be perhaps very short in the morning of a work day and be more detailed later in the evening, while on weekends the news can already be presented in more details in the morning. Thus, user models for personalization of multimedia content should be context-dependent and should be able to learn from user-system interaction in different contexts over time.

3.2 User Modeling and Profiling in the Amigo Scenario

3.2.1 Amigo Scenario Examples

In order to determine the basic architectural model and functionality of the User Modeling and Profiling Service, relevant scenario elements, as well as user needs and requirements, related to the scenario elements, have been identified.

1. Default and Initial UMPS Settings

Stereotypes: parent, child, friend, guest
Initial Profiles: Maria, Jerry, Roberto, Pablo, John

2. Profiles Creation and Adaptation

Scene 2, element 9

Amigo functions as an usher-service for Maria and Jerry. This service *recognizes people at the front and patio doors of the house and opens the door(s) for them if Maria and Jerry have authorized them*. Amigo, like a real professional usher can notify the inhabitants, can take visiting cards and show personalized marks of the people that are present in the house, for example, a picture, a note, light or a color code.

3. Users Identification and Verification

Scene 1, element 7

When Roberto's friends are coming over to his house for video gaming, Amigo downloads their profiles and integrates their game devices.
Scene 2, element 9 (see previous section)

4. Content Personalization

Scene 1, element 1

Maria's Amigo system works throughout the day for her. When she wakes up in the morning, *Amigo starts her day with playing music from her preferred play list, her favorite radio host, showing her personalized news or general summaries of hot news topics*. Amigo can order the program such, that it follows her everywhere in the house.

Scene 1, element 5

Amigo is Roberto's playmate for video games. *Amigo keeps track of Roberto's play list, the status of his scores, and the people he has been playing with, downloads the games*, and protects him from inappropriate games and content after consulting this with one of his parents.

Scene 2, element 12

Amigo maintains the overview of the food and household stock and generates shopping lists at predetermined time intervals. *The shopping lists are personalized*, but they take items that are on special offer, seasonal variations and nutritional balance into account.

Scene 3, element 14

Maria and her father have continued their habit of exercising together. Amigo sets up their exercise bikes, maintains their training schedules and lets them cycle through the video landscape displayed on the video wall in Maria's home (the one used by Roberto for games). John watches the same scenery on his display, they see each other working hard, *but Amigo makes sure that they each have their personalized exercise program*.

5. Multi-User Adaptation

Scene 1, element 3

Amigo does the same for Jerry. Maria and Jerry can each have personalized news and entertainment programs at the same time independent of their location in the house. If Amigo is following them and *if they happen to be in the same room, or with other members of the household, Amigo discretely stops and waits for new orders*.

Scene 3, element 13

Roberto and his grandfather John have continued their habit of playing games together, watching a bit of TV and having their man-to-man chat. Amigo takes care of setting up the right ambiance. John's Amigo system is a modest version, with which he can be a participant in Roberto's games, just like Roberto's peers. *Amigo selects the games that both John and Roberto like.* They can look at each other and see what game moves are being made. Amigo can also set-up a video-conference for them in which *they can watch TV together*, show the newest acquisitions of their collections, or just tell their stories. With Pablo, the little one, John plays 'hide-and-seek' via this video communication.

6. Environmental Conditions Adaptation

Scene 1, element 5

When light and sound effects are important for Roberto's video games, Amigo adapts the home environment to the game by presenting lights, sounds and video throughout the room. Roberto is then in the center of the video game and can play by moving his body. The video is presented on a large wall, which also can show the people with whom he is playing or that are on-line and want to participate.

7. User Interfaces Adaptation

Elaboration of Scene 3, elements 13, 14 and 15

(Interface Adaptation is part of every explicit interaction in every scene of every scenario. To point out the adaptation, a new scene is proposed, in which the main characters have to face another Amigo system).
When visiting her father in Brussels Maria is able to control her fathers home the way she is used to. Amigo downloads her profile and adapt to her preferred vocabulary and structure. So, she has no problems to navigate and call the right functions to control devices while helping her father preparing dinner. Meanwhile Roberto is watching a movie. He has no problems to use his grandfathers Amigo entertainment devices, because the Amigo user interface has adapted to his personal profile.

3.2.2 Related User Requirements

User and technical requirements, extracted from WP1 results, are presented below:

3: Enable individual settings and preferences

16: The system should provide concurrently the appropriate information to the right persons for the appropriate occasion at different locations, i.e., filter information, provide resumes, according to user preferences (note people refer to existing services that they know)

17: The system should enable easy access and usage of information and data from different sources

22: The system should be energy saving

24: The system should maintain the appropriate environmental conditions of the house (temperature, humidity, light, air, dust, mites, etc.)

25: The system should support the activity organization and planning for multiple persons at home, between homes and between home and work

28: The system should support alignment of individual and group planning, updates and notifications.

30: The system should support the integration of playing computer games in family routine, and approved settings.

31: The system should support playing games and entertainment with multiple people in the same room or networked environment.

32: The system should take implicit social rules of behavior into account

35: The system should support keeping in touch with select group of friends, no need to always be connected as “me”-time is just as important.

36: The system should support feeling of connectedness to family and friends

37: The system should support ‘trusted’ relationships, e.g. meeting new people mainly through mutual friends.

3.3 User Modeling and Profiling Service Architecture

3.3.1 Introduction

The User Modeling and Profiling Service (UMPS) provides the methodology to enhance the effectiveness and usability of services and interfaces in order to (a) tailor information presentation to user and context, (b) reason about user’s future behavior, (c) help the user to find relevant information, (d) adapt interface features to the user and the context in which it is used, (e) indicate interface features and information presentation features for their adaptation to a multi-user environment. Constructing, maintaining and exploiting user models and profiles, which are explicit representations of individual users preferences, achieve these goals.

As an example, the system performs context-based multimedia personalization by using the UMPS to retrieve multimedia objects which the user is interested in or likes, reducing the time required to browse through, (e.g., a large video collection or the complete news record) and to personalize the presentation of these objects (e.g., duration of news presentation, language and the need for subtitles for video presentation, sound level). For achieving this goal the system needs to recognize user’s context (time of the day, number of persons in a room, etc) and to select the appropriate system’s action in this context (e.g., to stop news presentation if both spouses are in the same room).

Personalization can be achieved either in completely autonomous mode, when the system decides what to present to the user, or in a semi-automatic mode, when the system makes suggestions to the user. In the autonomous mode, the system creates, for example, news' digests corresponding to the user model and presents these digests. This way is the most comfortable for the user, provided that the system has a very good user model. However, if the model is not good enough, the system can make mistakes (e.g., not informing the user about the voting results for the EU constitution because the user did not care about such issues earlier), reduce the trust in the system, and irritate the user. In case of the semi-automatic mode the system makes suggestions to the users (for example, presents the short list of top-ranked videos and asks which one to show today) allowing them to feel in control, which they usually like, and which should reduce the number of system mistakes. The disadvantage is the increased need in user attention compared to the fully autonomous mode. These two approaches are not exclusive: in one context it is appropriate to act autonomously and in another context it is better to ask the user's opinion. However, at earlier stages of user-system interaction, when the user model is not well known yet, the semi-autonomous mode would be more preferable.

Thus, in a first step, a user model is built, which enables the system to provide the user with a reasonable list of suggestions for retrieving multimedia objects, e.g., to suggest a list of new videos or headings of news for this evening. The next step is to refine the user model based on the interaction/context history, the choice made by the user from the system's suggestions, and the feedback of the user, provided that the user agrees to give feedback (e.g., to answer, whether the user liked the video or not). The update of the user model by learning from interaction history is a substantial research problem since the users are generally not willing to provide feedback and their interests change over time.

3.3.2 Conceptual Data Model

3.3.2.1 Stereotypes

A stereotype is a collection of preferences that can describe a certain group of users of the system and may range from very general to very specific ones (i.e. adult, male, medical doctor, catholic). The stereotypes are arranged into a directed a-cyclic graph formed by the partial ordering relation "generalization of", allowing information not to have to be represented identically in many different stereotypes. The most general node of any stereotype structure is the stereotype ANY_PERSON.

Each preference in the stereotype represents a "key", and has associated a "value", and each pair of "key"-"value(s)" has an associated rating. The rating is a representation of how confident the system is that the "key"-"value" pair to which it is associated is correct for that group of users. In addition, the potential use of associated context to each key will be investigated.

The stereotypes can be considered as the common sense knowledge of the system about its potential groups of users, representing valid assumptions normally true in normal conditions.

3.3.2.2 User Profile

The user profile is a tree-based representation of individual user preferences and personal data. Preferences and data are grouped, in agreement with knowledge representation in the system. An example is the group of settings and preferences that define the user model needed for personalization of a certain interface (i.e. speech-based).

Initially, the profile is built by combining the preferences from activated stereotypes with explicitly acquired preferences. Following, user preferences are updated, or new ones are added, by dynamically modeling the feedback obtained from user-system interactions. For activated stereotypes, each "key" – "value(s)"-"rating" triplet has also associated a justification (e.g. from which stereotype it was obtained) and possible additional parameters (e.g. "context(s)" in which the preference is valid). The triplet "key"-"value(s)"-"parameter(s)" (e.g., "weight", "confidence", "context(s)") is the base for representation of settings in the profile tree.

3.3.2.3 Feedback Data

Feedback data is obtained from application services or the Intelligent User Interfaces Services, as recordings of user-system interactions and as implicit or explicit user feedback with respect to user's preferences. These data are composite structures of information, and may vary from recording of very simple user preferences, to preferences accompanied by complex contexts in which they are valid. Feedback data can be considered as the knowledge of events and facts, from the real-time/history record of the system with respect to a certain user.

3.3.3 Design Overview

Figure 3.1 illustrates the architectural model of the UMPS. The UMPS has an add-on modular architecture. The basic inner shell of the architecture is the *Core Profile Service* (CPS).

This service handles the profile information storage, the request-response operations to other services and applications layer, the information flow to the middleware, as well as the security issues. At the CPS level, the update of the profile will be based on static modeling methodology.

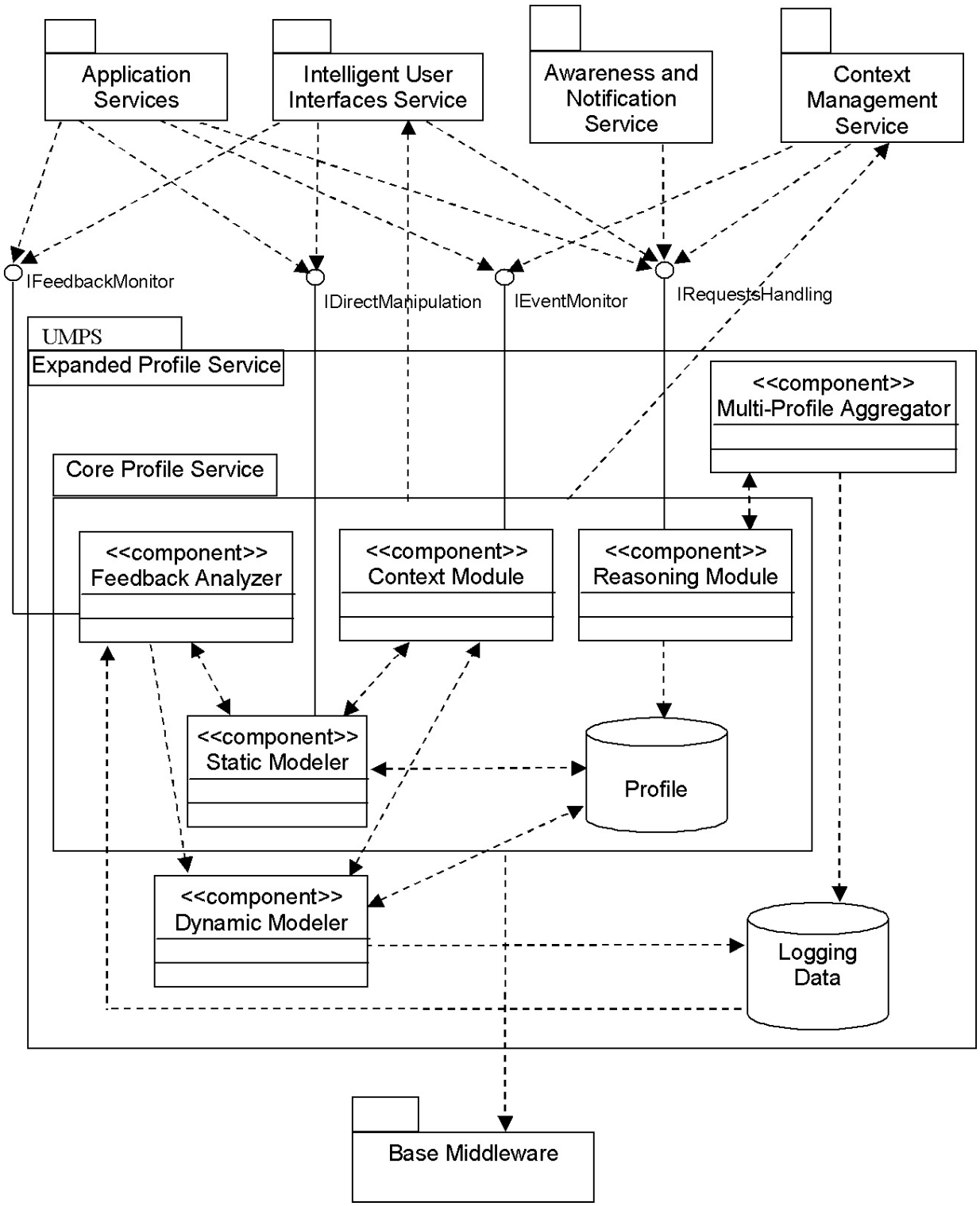


Figure 3.1 Architectural model of the User Modeling and Profile Service (UMPS)

The major components of the CPS, providing the functionality necessary for the interfaces offered to other system services and to the applications domain, are: (a) the *Reasoning Module*, (b) the *Static Modeler*, (c) the *Feedback Analyzer* and (d) the *Context Module*.

The *Reasoning Module* component is responsible for exploring the user profile and responding to other services requests needing either parts of the user profile (collection of user preferences for a particular situation) or discrete preferences.

The *Static Modeler* is responsible for the creation, removal and modification of user profiles at the user or application's request. The user may set either one preference at a time or a series of preferences corresponding to a branch in the user profile; and the Intelligent User Interface Service should support this.

The *Feedback Analyzer* will enable the update of the profile at system's initiative, based on explicit or implicit user feedback. In case of explicit feedback the user answers to well defined system questions, while for implicit feedback meaningful semantics are automatically extracted by the system from user's comments. For example, after a session with the user watching a video, the system may ask «Did you like the video?" and according to the answer update the related profile settings: check if the movie type is in the profile, if not add it along with the user's feedback, else update (increase or decrease) value of user interest in this type of movies. Implicit feedback is selected by the system without explicitly asking the user. Considering the previous example (user watching a video) and supposing that the system suggested the video, if after 10 minutes the user requests, "Please change the video, I don't like it." The system must consider this implicit feedback for the next video suggestion.

Context Module. This component receives «context changed» events, and requests the new context information whenever necessary. If the current context was already recorded in the user model, but the user's preferences are changed, then the UMPS updates only the settings for this context. If the context is new, then a new record should be added to the user model.

The *Expanded Profile Service* (EPS), the outer shell of the UMPS, will implement enhanced functionalities of the service, including the *Multi-Profile Aggregator* and the *Dynamic Modeler*.

The *Multi-Profile Aggregator* component provides an aggregated profile in case of multiple users found in the same context (i.e. the same room).

The *Dynamic Modeler* is responsible for the modification (update) of the user profile using the logging data, resulted from implicit or explicit user feedback. This update can be made either by activating additional stereotypes, as a result of triggering events, or by learning of new parts of user profile in a different way, e.g., by training a classifier to make decisions on whether the user will like some video or not in the current context.

3.3.4 Interfaces

UMPS offers four interfaces to the other services and applications of the system:

- *IRequestsHandling* interface (Figure 3.2 A), offered by the *Reasoning Module* to any other service or application of the system, to explore the user profiles.
- *IDirectManipulation* interface (Figure 3.2 B), offered by the *Static Modeler* to application and user interfaces services, to perform any direct operations on user profiles.
- *IFeedbackMonitor* interface (Figure 3.2 C), offered by the *Feedback Analyzer* to application services and the Intelligent User Interfaces Service, to monitor any implicit or explicit feedback events.
- *IEventMonitor* interface (Figure 3.2 D), offered by the *Context Module* to context providers, to indicate context changes.

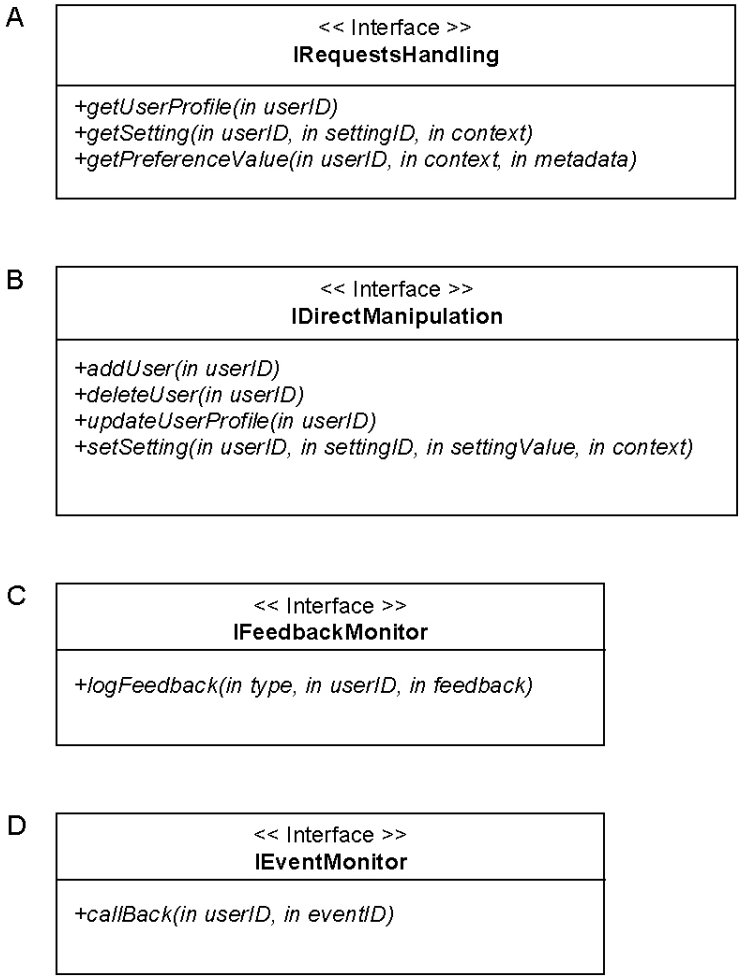


Figure 3.2 User Modeling and Profile Service (UMPS) Interfaces.

The UMPS provided capabilities, offered through the previously mentioned interfaces are:

- *IRequestsHandling::getUserProfile(in userID)*: provided capability for the exploration of the user model
- *IRequestsHandling::getSetting(in userID, in settingID, in context)*: provided capability to retrieve the value of a certain user preference for a given context
- *IRequestsHandling*::getPreferenceValue(in userID, in context, in metadata): provided capability to retrieve the estimation (rank) of how the user will like the available video or game in a given context
- *IDirectManipulation::addUser(in userID)*: provided capability to build a new user profile at user's or application's request
- *IDirectManipulation:deleteUser(in userID):* provided capability to remove an user profile
- *IDirectManipulation::updateUserProfile(in userID)*: provided capability to update a profile at user's or application's request
- *IDirectManipulation::setSetting(in userID, in settingID, inSettingValue, in context)*: provided capability to allow direct modification of discrete preferences

- *IFeedbackMonitor::logFeedback(in type, in userID, in feedback)*: provided capability to keep the history of user preferences in given context, extracted from its implicit or explicit feedback
- *IEventMonitor::callBack(in userID, in eventID)*: provided capability to receive context changed events

3.3.5 Dependencies

The UMPS depends on the Context Management Service, to retrieve context following to a "context changed" event notification. Also, UMPS depends on the Intelligent User Interfaces Service, to use the interfaces for direct manipulation of profile settings and to request explicit user feedback.

In addition, UMPS depends on the Amigo Base Middleware, which should provide functionality for registration and discovery of services, data storage and retrieval, and security as described in Chapter 1.

3.3.6 Dynamic Behavior

An example of UMPS interaction with the other services in the Amigo system is presented in Figure 3.3. In this example, the following scenario is considered: Roberto's friend is coming over to his house for video gaming. The Intelligent User Interfaces Service identifies the friend, and in order to verify its identity asks UMPS for its voiceStamp and model. Once the user if verified, Intelligent User Interfaces Service will log to UMPS the last utterance for future model update (and this may be done in different situations important to have an accurate model of the speaker). The Intelligent User Interfaces Service sends the identity of the user to the application service for further actions. The application service will get the current context (Roberto is in the house) and will request from UMPS the access rights of Roberto's friend in the current context (a notification may be send to Roberto, to inform him about his friend's arrival). When the two of them are in Roberto's room, the Context Management Service informs UMPS about the context change (two persons in the same room). UMPS will perform a *multiprofileAggregation*, to find the best environment setting in this context (light intensity, music volume, etc.). And these settings are returned when requested by the application. Next, Roberto's friend will select the game. The Intelligent User Interface Service will present, at first, the games from its profile. However, he decides to select a game he never played with Roberto. When the application gets informed about the selected game, it will request UMPS to log the selected game. The *Feedback Analyzer* may decide to update the profile (*dynamicModeling*), as the game is new. At the end of the game the application may ask UMPS to save the scores and any other game related data to their profiles (eventually, by asking them in advance if this action is wanted). UMPS finds that the game is new for Roberto's profile, and will send a request to the Intelligent User Interfaces Service to get some explicit input on how much Roberto liked the game. It will also update its profile, to get the new data into consideration.

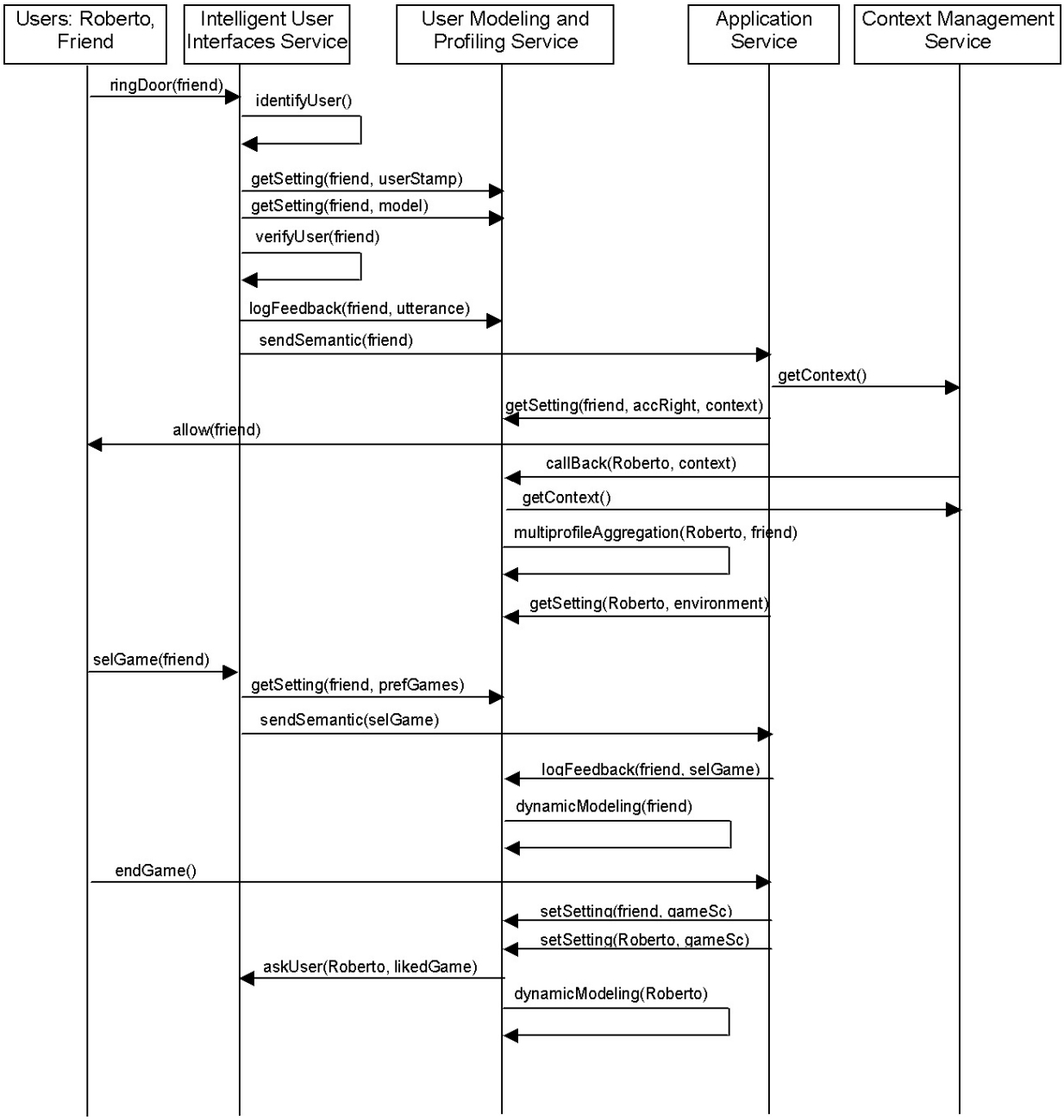


Figure 3.3: Example of UMPS interactions with other services.

4 Awareness and Notification Service

4.1 Introduction

The main function of the Awareness & Notification Service from the system viewpoint is to make application layer services aware of any significant change in context, by notifying them as soon as such a change occurs. A major focus from the user perspective is on rendering changes in the availability, status and behavior of users and applications such that users are kept aware of each other's availability and can share this awareness information.

The Awareness and Notification Service exploits information provided by different sources, either in the form of basic data obtained from sensors and user interaction, or as data already aggregated at a higher level. The service provides basic functionality that is needed to develop computer-mediated communication systems that allow small intimate groups of users to stay aware of the activities and presence of their peers, with minimal effort, over prolonged periods of time.

A crucial aspect for the Awareness and Notification Service is to provide notifications that take the social situation of people and the urgency (or content) of the information into account. It should also be able to select the appropriate means of presentation, adapt the level of rendering intensity (e.g. light vs. loud sound in the case of an audio based notification), and distinguish between the receptors for providing the notification. The Awareness and Notification Service is instrumental in showing that an ambient system is aware of behaviors of people and devices. If the ambient systems detect deviations from the usual pattern of behavior of a person, a small group of people, or of devices, and makes inferences about the changes in, for example physical and social context, the Awareness and Notification Service makes people aware of this information when and where appropriate. The Amigo scenario (Amigo Deliverable D1.2 [2]) contains many examples in which the Awareness and Notification Service plays an important role in making people aware of each others' whereabouts in their social/physical contexts.

4.2 Awareness and Notification in the Amigo Scenario

4.2.1 Amigo Scenario Examples

The following elaborations of the Amigo scenario illustrate the role of the awareness and Notification service

1. <u>Action-Reaction</u>

This elaboration is based on Scene 2, element 9 of the Amigo scenario.

> Maria enters her apartment and slams the front door. She is in a hurry since she has to pick up Robert from the cinema in an hour. The Awareness and Notification Service recognizes her presence and detects that she shut the front door more powerfully than usual. It also detects that she is moving slightly faster than normal. The system concludes that Maria is present and is in a hurry.

Elaboration based on Scene 2, element 9 and Scene 3, element 13 of the Amigo scenario.

> Maria's father John lives far away from her. In his house an ambient display unobtrusively indicates that Maria is at home and that her activity level is high. He wants to give Maria a call. He glances at the ambient display and decides not to call her since he wants to talk to Maria without too much rush. Instead, he touches the ambient display in order to indicate to his daughter that he is thinking of her.
> Before leaving, Maria passes the ambient display in her house and spots her father's message. When she returned home from the cinema, she calls him.

2. I'm here Scenario

Elaboration based on Scene 2 element 9 and Scene 3, element 13 of the Amigo scenario.

> Maria returns home from work and the system senses her presence. The ambient display shows her presence and the availability of her family and friends. Maria also indicates her availability to her family via her personal mobile device.

Elaboration based on Scene 3, element 15 of the Amigo scenario.

> Her father notices Maria's availability and requests a video connection. When Maria starts a video connection, the system automatically switches her status to "present but not available". Since the conversation will probably take longer than expected, Maria uses her mobile device to exclusively signal her availability to Jerry. The ambient display shows that Jerry is at home, but only available in urgent cases, so Maria assumes that he is still working in his study and continues her conversation with her father. After some time, Jerry leaves his study and switches on the music in the living room. The ambient display shows that Jerry is now available. Maria says goodbye to her father and calls Jerry.

4.2.2 Related User Requirements

The following most relevant user and technical requirements from the list in Amigo Deliverable D2.1 [2] for the Awareness and Notification topics in the scenario elements are:

> 25: The system should support communication between different contexts, such as home <=> home, home <=> mobile user, home <=> public place, mobile user <=> public place, etc.
>
> 34: The system should enable communication with multiple people at the same time.
>
> 35: The system should support keeping in touch with groups of friends and family.
>
> 33: The system should support multiple (virtual) identities to allow interaction with different groups of people (family, colleagues etc.).

4.3 Awareness & Notification Service Architecture

4.3.1 Introduction

The main function of the Awareness & Notification Intelligent User Service is to make application layer services aware of any significant changes in context, by notifying them as soon as these changes occur. Standard awareness systems could be viewed as a specific use of this function, where context of interest concerns user availability and status, and where application behavior is affected by displaying or, more generally, rendering these changes so that users are kept aware and share the availability of each other. Additionally, it should be possible to notify different groups of people, such as family or colleagues.

The Amigo Awareness and Notification service corresponds highly with Event Notification Services (ENS) like Elvin, Sift, Siena and OpenCQ (Hin03 [14]). The main sub functions included in such services are the registering of "monitoring rules" of application services (i.e., rules that specify what context changes should be notified to the user), the evaluation and monitoring of context changes and the corresponding event triggering. Generally, the usage flow of this service is executed in 6 phases:

1. Definition of monitoring rules by the user/service.
2. Mapping of the user defined monitoring rules to the monitoring rule specification.
3. Invocation of the Awareness & Notification Service with the rules specified in the rule specification.

4. Discovery of correct context sources needed to evaluate the user rules by the Awareness & Notification Service.
5. Subscribing of the Awareness & Notification Service to the found context sources.
6. After a rule is triggered, the user or possible other parties must be notified.

For example, consider the “I’m Here” Scenario. Maria defines in her application that she wants to be notified of the availability of her family when she is present in her home. This rule is mapped to a rule specification language, for instance

`‘if <atHome> & <available(Family)> then [notifyMe]’.`

This is submitted to the Awareness & Notification Service, which searches for the right context sources to use when evaluating if Maria is at home and if her family is available. The Awareness & Notification Service subscribes to the context sources that can deliver this information and starts monitoring their context state. In parallel it monitors the inputted rule and if the condition part holds it triggers the notification to Maria.

4.3.2 Conceptual Data Model

4.3.2.1 Generic Rule Specification

The Awareness and Notification Service should notify persons based on triggered user-defined rules. A commonly used rule specification mechanism is event-condition-action rules, for example, ECA rules in (Doc05 [15]) and (Ipin01 [16]). They define a condition in which a rule is valid. This condition is a Boolean combination of several input events. Whenever a rule evaluates to true the action part is executed. A commonly used mechanism to define these rules is: (rule): if <condition> then [action].

For example, a rule “When I am in the city center and a buddy is also in the city center, notify me and my buddy”, could be expressed like this:

(buddy) : If <Me.inCity> & <Buddies.inCity> then [Me.Notify] & [Buddy.Notify]

Desirable aspects of such a language are:

- Expressive power: The language should be able to specify complex event conditions. For instance, combination of events with logical operators (AND, OR, NOT, etc). Simple event conditions could be expressed with a syntax close to context models.
- Convenient use: The language should be easy to use by application developers and users.
- Extensibility: The language should support definition of predicates and properties of these predicates on demand.

The rule triggering mechanism can have multiple characteristics:

- Time-based: The action part of a rule is triggered after a certain time. For example, notify me every 5 minutes on the location of my buddy.
- Change-based: The action part of a rule is triggered after a change has occurred. For example, notify me when my buddy’s location changes.
- Criteria-based: The action part of a rule is triggered on a certain criteria. For example, notify me when a buddy is in the city center.
- Hybrid: The action part of a rule can have a combination of the previously mentioned mechanisms. For example, notify me of my buddy’s location every 12 hours or when he/she is in the city center.

4.3.3 Design Overview

In Figure 4.1 the internal architecture of the Awareness & Notification Service is presented. The process starts with the parsing of the inputted user rules by the *Parser* (expressed in the rule specification language). The relevant events are extracted from the condition part of the rule. The *EventMonitor* finds and subscribes to the relevant context sources, which can deliver these events. If context sources cannot deliver context using a subscribe-notify mechanism (e.g. it only supports a request-response mechanism) this is implemented by the event monitor, which emulates the subscribe-notify mechanism, by implementing a polling scheme. If an event is triggered, this is passed to the controller. After parsing of the inputted rule, the *Controller* loads these rules and stores them in the collection Rules. It evaluates the loaded rules when an event change (received from the event monitor) has occurred. If a rule turns true (taking the security settings of involved parties into account) it executes the action part, which can consist of a notification of the user or other user and/or a change of modality of the user.

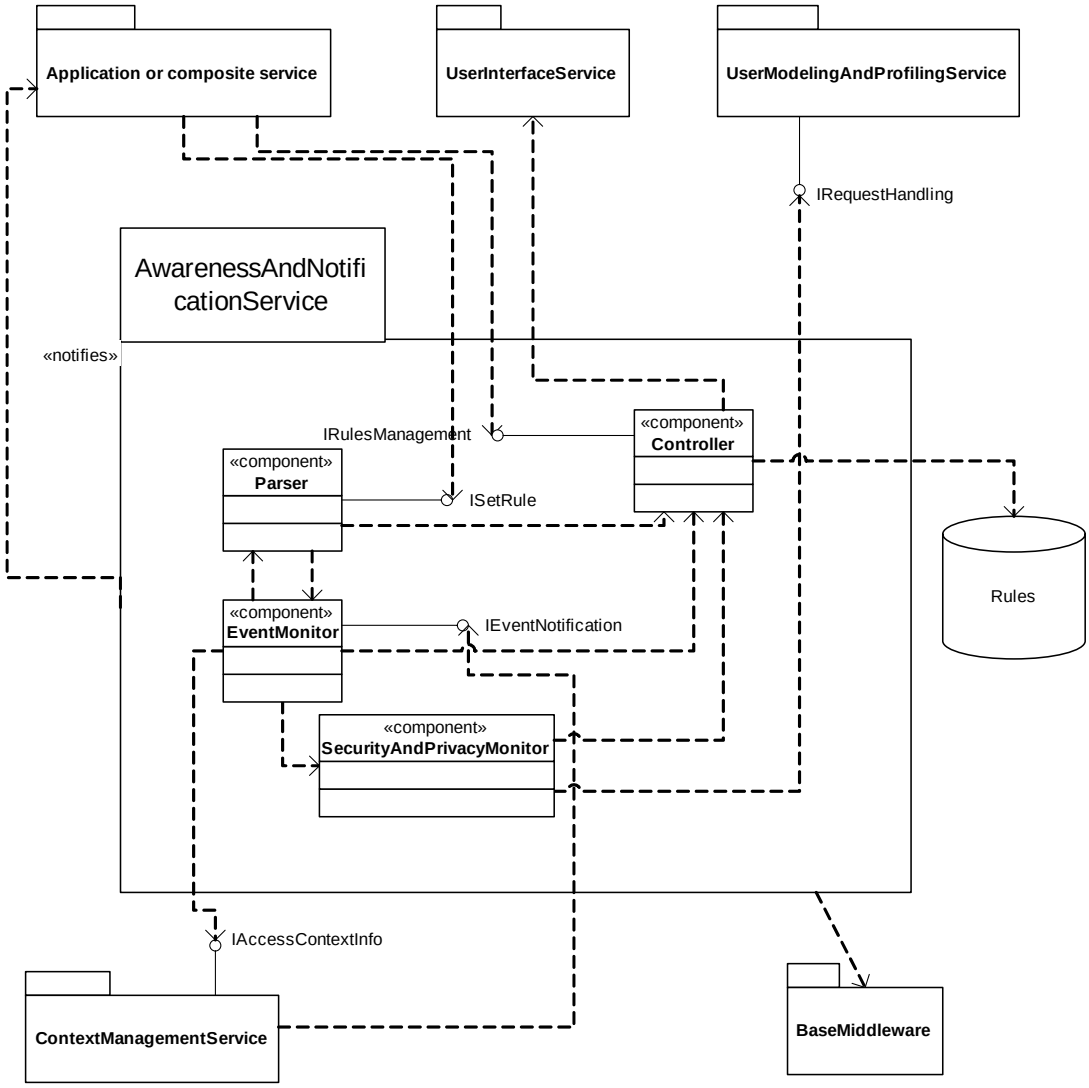


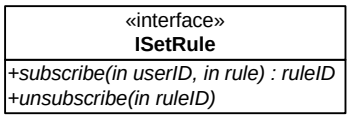
Figure 4.1: Awareness and Notification Service Architecture

4.3.4 Interfaces

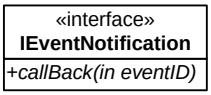
The architecture distinguishes three internal interfaces that are offered by the Awareness & Notification Service:

- Subscription interface offered to the application services (*ISetRule*)
- Event notification interface offered to the context providers (*IEventNotification*)
- Management interface offered to the application services (*IRuleManagement*)

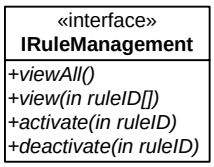
The *ISetRule* interface offers an interface to application services to be able to subscribe and unsubscribe to notifications by inputting rules into the Awareness & Notification service.



The *IEventNotification* interface offers a way for the context providers to call back the awareness and notification service to indicate a context change.



The *IRuleManagement* interface offers a way for the application services to manage the inputted rules. This includes viewing, activating and de-activation.



4.3.5 Dependencies

The Awareness and Notification Service provides the following interfaces to other Amigo services and components: *IrequestHandling* and *IaccessContextInfo.*

Awareness and Notification has several dependencies with other Amigo Intelligent User Services and the Amigo Base Middleware.

- This service uses the Amigo Base Middleware for service discovery, generic security and communication with other services.
- It uses the Context Management Service for retrieving context parameters to evaluate conditions in defined rules.
- It uses the User Modeling and Profiling Service for retrieving user information and preferences if needed to evaluate the rule or invoke the action.
- It uses the User Interface Service for displaying notification information for the rule creator or other users specified in the rule by the creator.
- It uses privacy and security for determining if it can retrieve context and user information and if it may invoke actions.

4.3.6 Dynamic behavior

Figure 4.2 presents an example of the dynamic behavior of the Awareness and Notification Service. In the diagram, we depict two phases of this service, the configuration phase and the operational phase. The scenario starts with the configuration of the service by Maria. She specifies that if she approaches her doorstep that the door should open and that her family should be notified of her arrival. Furthermore, she wants to know the availability of her family. The service instantiates this rule in the operational phase.

For the service to evaluate the defined condition, it subscribes to the needed context sources. In this example, that is the location of Maria. Her location is monitored until the expression turns true. Then the action part is executed. First, the needed services are discovered. In this example, the door service and the family availability service. First, the door is opened for Maria and then her family is notified. Then, the availability of her family is retrieved and presented on the appropriate user interface.

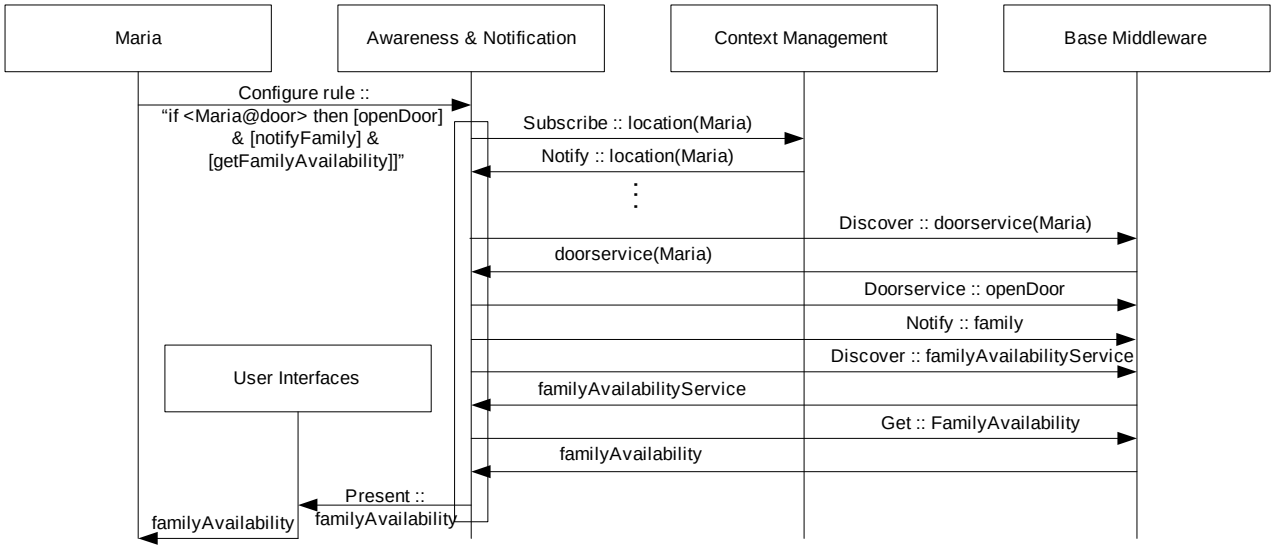


Figure 4-2: Dynamic behavior example

5 User Interface Service

5.1 Introduction

As humans perceive and communicate through several senses (e.g. touch, sight, speaking & hearing), the more modes human – system interaction involves the more friendly and human-centric the system might be perceived. Multimodal interaction, i.e., the combination of speech, gestures, pen and other, has become one of the driving factors for user interface technologies, since it allows to combine the advantages of diverse modalities and to compose more accurate information from the partially-complete data that are captured by the individual modes.

The User Interface Service provides means for services and applications to present appropriate interfaces to the users in an Ambient Intelligent environment. It provides a framework to manage explicit and implicit user interactions and presents the content to the users while supporting multiple interaction modalities. The goal is to support user interaction with multiple smart artifacts based on intuitive and natural interfaces that are multimodal. It consists of several sub-services that each tackle an aspect of the complex user interface service. There are concrete services that provide multimodal interaction functionalities such as the speech-based interface service or 2D and 3D gesture recognition services. A multimodal interface service is responsible for modality fusion to allow for truly multimodal interaction. A Multimodal Dialogue Manager (MDM) manages the flow of dialogs that an application requests. Finally, a Multi Device Interface Service provides the part of the User Interface Service that can be queried with properties of a given task and the available interaction devices returning suggestions on how to use which device in this situation.

The User Interface Service handles input/output and interaction devices as well as interaction modalities and copes with explicit as well as implicit user interactions. The dialogue component of the Amigo platform takes into account semantic considerations by proposing new standard semantic representation formats; multimodal fusion will further be based on this representation. The user interfaces designed for Amigo will support dynamic adaptation to context and devices and natural user interactions through multimodal (speech, 2D and 3D gestures) user inputs.

5.2 User Interfaces in the Amigo Scenario

5.2.1 Amigo Scenario Examples

This section illustrates the various components of the User Interface Service in an example derived from the generic Amigo scenario specifically tailored towards the interaction design in an Amigo environment. The example scenario is structured in two parts that can each be demonstrated independently or jointly as one large demonstration (Figure 5.1).

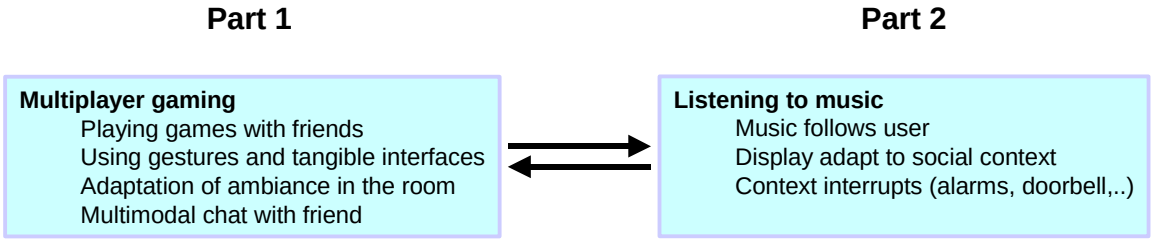


Figure 5.1: Parts of the User Interface Service example

1. Multiplayer Gaming

The following scenario description is an expansion of Scene1, elements 5, 6, and 7 and Scene 3 element 13 of the Amigo scenario.

This part demonstrates a parallel gesture interface for co-located gaming and the integration of a dedicated gaming device as well as the adaptation of the entire room to the gaming ambience. It further demonstrates how the two players can chat together in a multimodal and unobtrusive way, where both explicit and implicit interactions are used simultaneously. This part will be either scripted or used in a free-form mode allowing users to try out the game. It can be triggered by verbally commanding Amigo to turn on the multiplayer game.

Roberto and Peter enter Roberto's room.

Amigo senses the gesture device that Peter brought along with him. It displays images of available games on the wall display.

Roberto: Hi Peter, glad you could come.

Peter: Hello Roberto, what would you like to play?

Roberto: Well, pick a game yourself.

Peter navigates images back and forth by waving the control device in his hand (using gestures next, previous) and selects a game.

Roberto: That is a great game; let's start!

The lightning and ambient sound in the room adapts to the atmosphere of the game.

Roberto and Peter start playing, they control the game via gestures and by moving pieces on a physical game board.

When the game is finished, Roberto and Peter comment on their respective strategy and compare it with the previous games they had together a month before. The "chat-gaming" application follows their discussion in the background, and unobtrusively provides adequate feedback and information about the details of this old game (implicit interactions). Roberto and Peter can also explicitly interact, via voice and gestures, with these additional pieces of information, for example to focus on one of them and zoom it out on the screen.

2. Listening to music

This scene is an expansion of Scene 1, elements 1, 2 and 3 of the Amigo scenario.

This part demonstrates the music-follows-user functionality envisioned in the original Amigo scenario as well as a speech input interface and a multi-device interaction that changes interaction devices according to the current context. This part will be implemented as a scripted series of events that can be triggered by verbally commanding Amigo to turn on the Actor's favorite music.

Maria enters the kitchen and tells Amigo to turn on the favorite music.

Her favorite music is played on the public loudspeaker hidden in the ceiling.

She picks up clean clothes from the washing machine and carries them to the wardrobe in her room.

The music in the kitchen falls silent and now plays on the loudspeaker in her room.

While she puts her clothes in her wardrobe, she realizes that her ambient lamp on her bedside table starts to glow in a soft green tone, indicating that there is a message for her.

Since she has no TV in her room, Maria takes out a PDA and holds it near the lamp.

An Email has arrived and Maria starts reading it on the PDA. Since the Mail is important to her, she commands Amigo to lower the music's volume by saying "Volume down".

Because the Email is long and the text is hard to read on the PDA, Maria goes to the living room to read it on the large public display. The music follows her.

Upon entering the living room, the Email display moves from the PDA to the TV.

The doorbell rings. The music pauses.

Amigo: Peter is at the door; he is Roberto's friend. He wants to play a game with Roberto.

Maria: Please let him in.

When Peter enters, the music continues playing, but the Email moves back from the TV to the PDA.

Peter: Hello Maria, … uh that music is twenty years out of fashion.

Maria: Hello Peter, oh you don't like it? Amigo: I don't want to bother Peter with my music.

The music stops playing on the public loudspeaker and continues on Maria's collar.

Peter heads for Roberto's room, when he leaves, the music is turned on again on the loudspeaker and the Email is displayed on the TV.

5.2.2 Related User Requirements

User and technical requirements concerning User Interface Services, which were derived extracted from WP1 results, are presented below:

19: The system should support having control over data and information for best performance

20: The system should reduce the time needed for household chores and where possible do cleaning jobs

30: The system should support the integration of playing computer games in family routine, and approved settings.

31: The system should support playing games and entertainment with multiple people in the same room or networked environment.

32: The system should take implicit social rules of behavior into account

33: The system should support increasing number of communication moments in multiple different contexts

34: The system should enable communication with multiple people at the same time, e.g. broadcasting, democratic group planning.

35: The system should support keeping in touch with select group of friends, no need to always be connected as "me"-time is just as important.

36: The system should support feeling of connectedness to family and friends

37: The system should support 'trusted' relationships, e.g. meeting new people mainly through mutual friends.

5.3 User Interface Service Architecture

5.3.1 Introduction

The User Interface Service provides a framework to manage explicit and implicit user interactions and presents the content to the user supporting multiple interaction modalities taking into account the context information coming from the Context Management Service. The User Interface Services can be used for both explicit and implicit interaction.

For example speech and sound recognition can be utilized in activity detection, based on ambient sounds, making it possible to launch for example a picture slideshow application during a meeting. The Context Management Service can activate different UI services (modalities) to acquire contextual information from different modalities such as utilization of the speech service to find out if someone is speaking. Figure 5.2 presents the implicit and explicit communication between users, UI- and context services, applications and ambient environment, such as a home environment with ambient input/output devices.

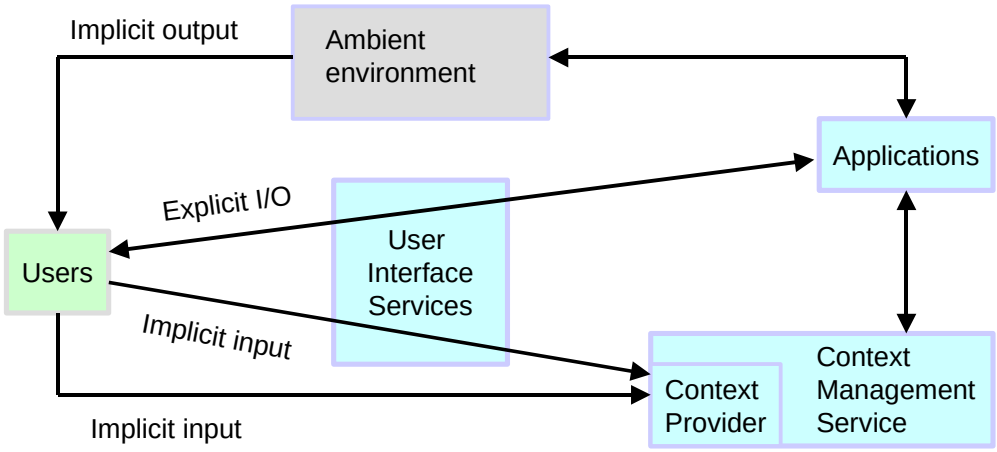


Figure 5.2: User Interface Service communication flow.

There are many different types of potential interactions between the user and the system. We propose to classify them in the three following classes: Natural Language, Body Language, and Direct Manipulation. This classification is mainly based on technical and engineering motivations: Each class actually corresponds to a type of treatment from the system point of view.

5.3.2 User Interface Modalities

Natural Language, Body Language, and Direct Manipulation are the main modalities that will be exploited by the User Interface Service. These modalities and how they will be exploited are discussed.

5.3.2.1 Natural Language

This class of interactions includes natural language sentences, either in oral or written forms. We exclude from these classes simple “keywords”; they don’t require any language-based processing and belong to the “direct manipulation” category. The sentences considered here may be spoken or written by the user. Different technologies may be involved to produce these sentences: speech recognition, optical character recognition, computer or mobile phone keyboards, etc. The Amigo architecture will support any of these interactions, but for demonstration purposes, we will focus on spoken language.

Voice

Natural speech communication, being the most appropriate single way, should be present in all the main rooms of the house and should work as a single interface for the main system. Microphones and loudspeakers should be present in all the main rooms and corridors, and the processing part of this “voice interface” may not be a part of the main system but an autonomous service.

The voice service consists of four main processing units:

- Signal pre-processing and acoustic scene analysis: Captures speech input and ensures that the signal is delivered as clearly as possible to the recognition process thus enhancing the performance of speech recognition. It also extracts and provides streams of metadata such as voice activity detection, speaker change detection and acoustic source localization and tracking.
- Speech input: Speech input is the unit where the system recognizes the user's speech, understands the meaning of the spoken utterance and acts accordingly. This unit includes the voice dialogue manager that is responsible for the entire speech interface coordination as well as for the communication with the application for transferring the required input or provided output data.
- Speech output: Generates and presents the system's response to the user. Depending on the current status of the interaction, this response may be a notification, a question, a confirmation, or the information requested by the user.
- Speaker recognition: Automatically recognizes who is speaking on the basis of individual information included in speech waves. This technique makes it possible to use the speaker's voice to verify their identity and control access to services and information.

The communication between the speech-based interface and the Amigo system (middleware services and Intelligent User Services) depends on the scenario: in a single-speech mode only scenario the interface directly communicates with the application that is responsible for integrating the middleware and the advanced services as well as the additional functionality in order to have an integrated functional environment. On the other hand in a multimodal scenario, modality-specific interfaces (i.e., voice service, gesture recognition service) are coordinated by a multimodal interface service that has the overall handling of all input/output data and communication.

5.3.2.2 Body Language

This class of interactions includes all the information that is conveyed via the human body, apart from the natural language that is treated apart. This includes, for example, gestures, facial and body expressions, and vocal paralinguistic information (laugh, shout, emotions, etc.). In Amigo, we will mainly consider the 2D and 3D gestures functionality, which consists of physical movement based interaction modules that can be used either as an individual control device or to supplement or augment other control modalities. 2D gestures are performed using tablet stylus (pen stroke recognition) and 3D gestures are detected using handheld device equipped with movement sensors. In addition the 3D gesture recognition module in Amigo has support for tilting and pointing detection.

2D Gestures

Gestures are essential to resolve references in a man-machine interface. The 2D tactile interface that we will use for demonstration in Amigo allows selecting of objects in a scene sorting them according to their saliency. The scene is compound of a background image and foreground artifacts. These artifacts could be either basic geometric shapes (e.g., circles, triangles, squares) or complex polygonal figures, each of them associated in the scene with a type and a unique identifier. The selection of these objects can be made with four types of tactile gestures: circling, targeting, pointing, and scribbling. The gesture is first digitalized and reduced for efficiency to its most significant points. Each object, which intersects with the gesture, can be a part of the selection. The salience of a selected object is then computed with the percentage of the object, which is in intersection with the gesture. A fully circled object will have a 100% saliency, as a half circled one will have a 50% saliency. The list of the selected objects (identifier and type) is finally outputted in the MMIL-format (Lan04 [17]) (MultiMedia Interaction Language) as a visual-tactile event (see section 5.3.3).

3D Gestures and Pointing

Free-form 3D gesture recognition is typically based on cameras or movement sensors. Here we focus on hand gesture recognition using a specific hand held sensor device developed by VTT Electronics (Kel05 [18]). The sensors on the device can be used to detect discrete pre-trained gestures based on acceleration patterns during gestures. In addition the sensor device supports the physical selection of objects by means of infrared-based pointing. This enables the user to first select a known target device from the environment by pointing, and then control it with a set of gestures. Small wireless infrared tags can be attached to different objects in the ambient environment, i.e., control targets such as home appliances, and selected by pointing the hand held gesture control device towards the object. Gesture movement data is wirelessly transmitted to a computer where the 3D Gesture Service takes care of data processing and recognition. The 3D Gesture Service provides methods to train and recognize personal hand gestures. The 3D Gesture Service provides the gesture, tilting and pointing events to Multimodal Service, which takes care of the integration and filtering of different modality inputs (see section 5.3.3).

Multimodal Fusion

The Multimodal Service in Amigo provides the ability to interact with the system by the joint channels of voice and gesture. Either modality (2D gestures, 3D gestures and voice) can be used for designating or pointing at some objects (physical devices, virtual objects or more abstract, discursive objects) and referring to actions to be carried out on them by the system. The aim of the Multimodal Fusion is thus to coordinate information expressed on these parallel channels and to fuse them into a unique representation of what the user requests. Although 2D and 3D gestures would generally provide the same types of information (selection of one or several objects and/or specification of an action to be performed), the technical apparatus used for 2D and 3D management are quite different and thus for complementary uses.

On the one hand, 2D gestures are usually performed with a pencil on a touch-screen. They are thus dedicated for the designation of virtual objects, displayed on the screen. Fusion will thus concentrate on analyzing the paths drawn on the screen (with a quite good accuracy) to identify objects and solve referential expressions on the voice channel (deictics, pronouns, etc.). On the other hand, 3D gestures will not actually be recognized as paths but as acceleration patterns. Those gestures cannot refer to objects (what 2D can do) but rather to predefined commands (e.g. play/stop, switch on/off, open/close, etc.). Fusion will thus consist of identifying which objects are concerned by a given gesture. Objects can be actual physical devices designated by infrared pointing or virtual/abstracted objects designated on the vocal channel.

Those two gesture interfaces are complementary, since 2D cannot directly address physical devices and 3D cannot address directly virtual objects projected on a screen or on a wall. However, it is worth noting that both modalities can address any of these types of objects after fusion with the verbal content.

5.3.2.3 Direct Manipulation

This class of interactions includes all the other types of interactions. They are mainly composed of manipulation of physical or virtual devices, hence the name of the class. For example, switches, sliders or buttons belong to this class. Other examples are selecting one or several items amongst a group, or choosing a real value in a range. Typical technologies involve widget-based graphical user interfaces (GUI), mouse, keyboards (but that are not used to type in sentences), pressure sensors, etc.

While the two previous classes of interactions originate from the user, this one is created by the application or the system. Hence, the semantic associated to the manipulated items is defined by its creator/manager, i.e. most often the application. But sometimes, the UI services or any other service may also use such interactions widgets to get simple inputs from the user.

Graphical user interfaces will also be used for user access and control of Amigo devices and services. Therefore, accessible functions of different Amigo environments must be discovered and dynamical integrated into a graphical user interface structure, which is adaptable to different environments and user preferences.

5.3.3 Conceptual Model

The core component of the User Interface Service is the *MultiModalService* (Figure 5.3) that manages the communication between other Amigo services and the different user interface I/O. Multimodal inputs and outputs can be categorized in natural language communication (voice), body language communication (gestures and pointing) and direct manipulation (keyboards, remotes, etc.). In addition, ambient interaction is inherently mobile and involves communication between different persons and devices in different places. The MultiDeviceService manages this. Figure 5.3 illustrates the conceptual architecture of the User Interface Service.

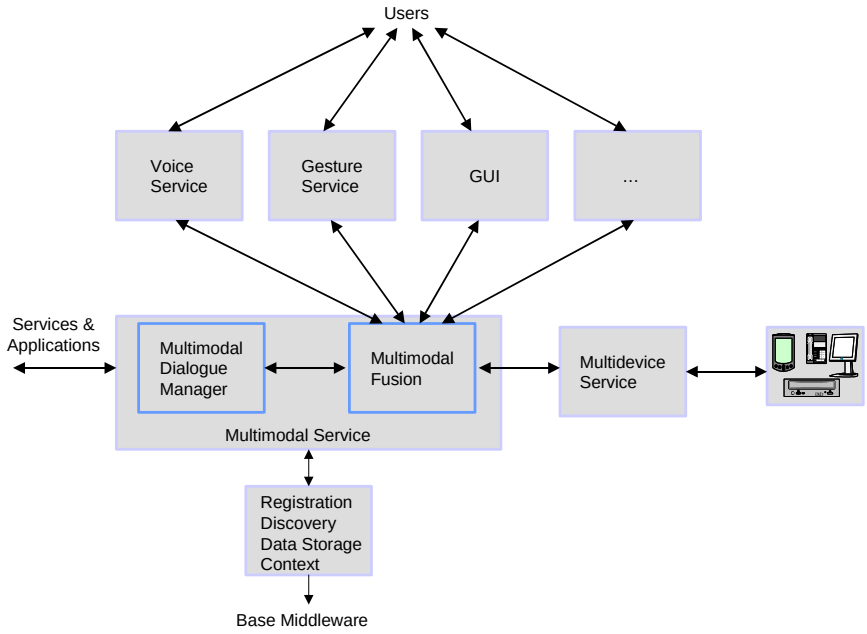


Figure 5.3: Conceptual model of the User Interface Service

The main components in the core architecture of the User Interface Service are: the Voice Service, the Gesture Service (2D and 3D), Graphical User Interface GUI, the Multidevice Service and the Multimodal Service that consists of the Multimodal Dialogue Manager (MDM) and the Multimodal Fusion Module (MFM).

Multidevice Service

In an ambient intelligence environment, interaction between users and/or ambient computer systems is manifold. Sometimes, only small pieces of information must be communicated privately from one user to a device. Sometimes, several users must collaboratively perform complex configuration tasks with multimodal interactions working at a group level. And sometimes, there is human discussion that does not involve and should not be disturbed by the ambient intelligence interaction devices at all. Thus, for different interaction needs, different media and modalities are more suitable than others (Mag04 [19]).

With the Multidevice Service we attempt to let the ambient intelligence environment assist in finding optimal compositions of modes and devices. By specifically addressing the uniqueness of different modes and devices, a dynamic, re-configurable, and context-aware composition of devices can be achieved.

The Multidevice Service assists in finding the optimal set of devices and interfaces by matching the interaction characteristics of each available device with the properties of each so called interaction request an application might formulate. Such an interaction request might be to present the user either privately or publicly an incoming email, depending on who else is around. For instance, if only family members are around, the large TV screen might be used to display the greeting card of the grandmother, but when strangers are around, the mobile phone of the recipient might vibrate and the reception of the mail might take place on the PDA screen, which is harder to read, but obviously more private than the TV. The actual devices and modes involved in a given game situation are determined considering several findings from modality theory, for instance that linguistic input and output is unsuited for specifying detailed information on spatial manipulation or that static graphic modes are suitable for representing large amounts of (complex) information. Thus, the Multidevice Service maps abstract requests from applications that inform about how the interaction should take place to the concrete modes and devices currently present and operational.

Multimodal Service

The Multimodal Dialogue Manager (MDM) coordinates the communication between the modality services (e.g., Voice Service, Gesture Service) and the Amigo services and applications. Individual modality services collect the input from the users by means of dedicated IO hardware such as microphones, motion sensor devices, etc. The input data is processed by the Modality Services (e.g., Voice Service, Gesture Service) and the resulting output is sent to the Multimodal service. On the other hand MDM contacts other Amigo services either to gather information required for the system-user interaction (i.e. the user profile from User Modeling and Profiling Service) or to transfer a user demand to be executed on a specific device (i.e. arrange the room lighting for a “watch movie environment”). The MDM uses internal dialogue models and application scripts to coordinate and synchronize the interaction between user, input modalities and application(s). The multimodal fusion (MFM) module is used to merge and filter all the partial interpretations.

Basically, each of the input-related modules produces partially context-dependent interpretation of the information coming from the modality they are responsible of. Each of these partially context-dependent interpretations may be made of several hypotheses. For instance, a gesture interpretation module is intended to produce a list of hypotheses on potential candidates to a selection gesture. As this information is made both from the gesture and from the knowledge about the object context, but doesn’t take into account speech or selection history, it is considered to be only a partially context-dependant interpretation. In order for the interpretation to be complete, the multimodal fusion (MFM) module has to merge all the partial interpretations.

Typically, merging this information means two things:

- Select appropriate hypotheses based on semantic compatibility information and
- Accept/reject fusion of under-specified semantic items based on temporal relations (for instance, gesture/deictic fusion is allowed only when the gesture’s start precedes the end of the utterance containing the deictic). These rules are rarely modality independent and must often be determined during an experiment.

As the MMF module has to take its inputs from several modules, and these inputs must contain both semantic information and specification of hypotheses, we plan to use an

interaction language that has been designed exactly for this purpose, the MultiModal Interface Language (MMIL).

From a data model point of view the MMIL structure is based on a flat representation that combines any number of two types of entities that represent the basic ontology of any dialogue system, namely *events* and *participants:*

- An *event* is any temporal entity either expressed by the user or occurring in the course of the dialogue. As such, this notion covers interaction event (spoken or realized through the haptic interface), events resulting from the interpretation of multimodal inputs or event generated by decision components within the dialogue system.
- A *participant* is any individual or set of individuals about which a user says something or the dialogue system knows something about. Typical individuals in a dialogue environment such as Amigo will be the users, multimedia objects, services and graphical objects. Participants can be recursively decomposed into sub-participants, for instance to represent sets or sequences of objects.

MMIL is extensible: as MMIL's meta-model is simply composed of three types: events, participants and relations, it is easy to extend it in order to take into account new semantic types. That way, a fusion module that can handle MMIL will be straightforwardly adaptable to new modalities or information sources. Also, MMIL is incremental. In an architecture where all modules communicate using the MMIL language, the added value of each module can be kept along the stages of processing. This allows all hypotheses to be retained and processed on later stages, which is necessary when disambiguation cannot be done in one step. MMIL is hence an interaction language that can be used at any stage in the multimodal interpretation process, and even between all the components of a multimodal dialogue system.

5.3.4 Design overview

In Figure 5.4 the internal architecture of the User Interface Service is presented. The main architectural components of the User Interface service are:

InteractionService manages the communication between other Amigo Intelligent User Services and the input and output modalities. Components within the *InteractionService* are the *VoiceService*, the *GestureService* and the *GUIConnector.*

MultiModalService, which consists of the *MultiModalDialogueManager* and the *MultiModalFusion* components. The *MultiModalDialogueManager* coordinates the communication between the Amigo services and the output of the components of the *InteractionService*, e.g., *VoiceService, GestureService.* It uses internal dialogue models and application scripts to coordinate and synchronize the interaction between user, input modalities and applications. The *MultiModalFusion* merges and filters all the partial interpretations.

DeviceService. The major component is the *MultiDeviceService*, which finds the optimal set of devices and interfaces by matching the interaction characteristics of each available device with the properties of each interaction request that is formulated by an application. The *MultideviceService* maps abstract requests from applications that inform about how the interaction should take place to the concrete modes and devices that are present and operational.

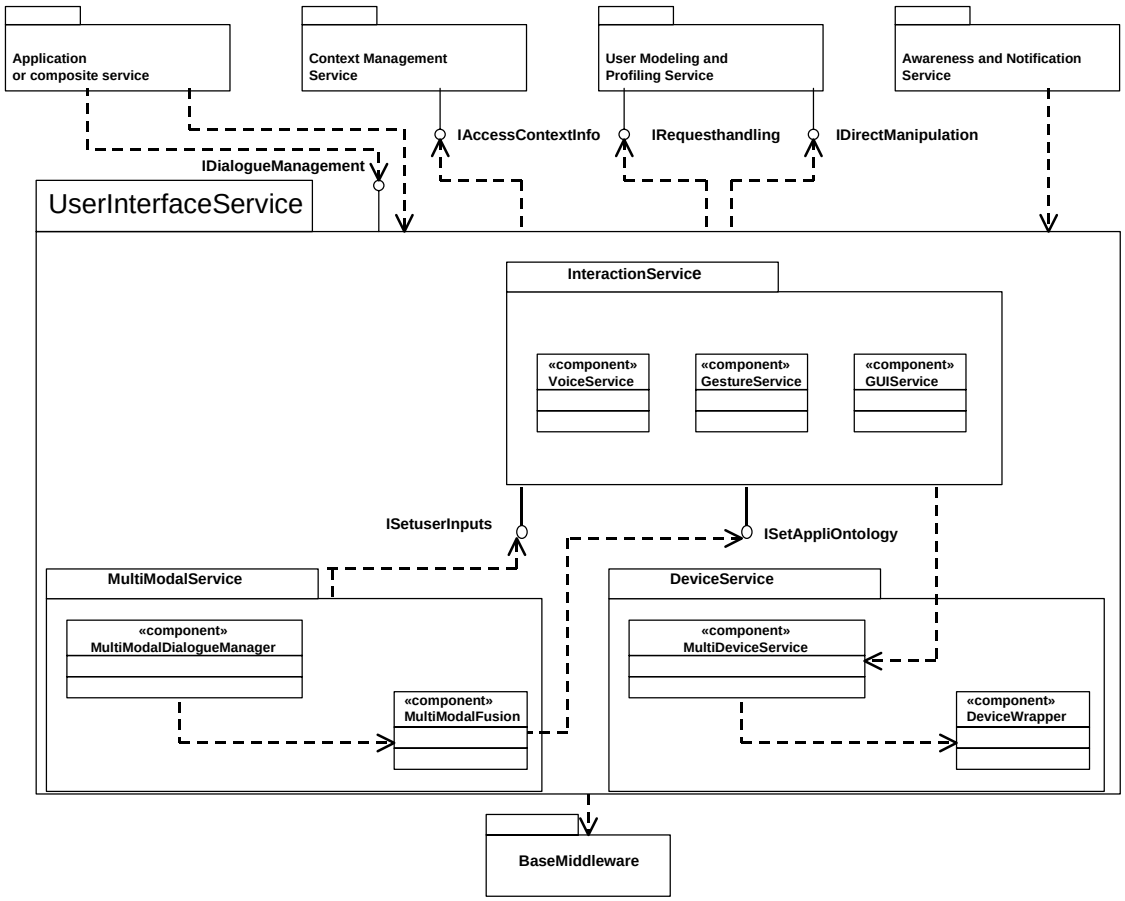


Figure 5.4 Static description of the User Interface Service including its interfaces and dependencies

5.3.5 Interfaces

Internal Interfaces

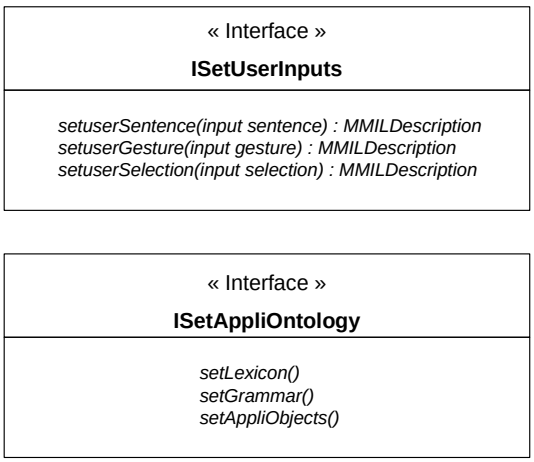


Figure 5.5: Internal interfaces

Figure 5.5 shows the two internal interfaces that are distinguished in the User Interface service, i.e., *IsetuserInputs* and *IsetAppliOntology*

5.3.6 Dependencies

The User Interface Service provides the following interfaces to other Amigo services and components (see figure 5.4): *IDialoguerManagement*, *IRequestHandler*, *IDirectManipulation* and the *IAccessContextInfo*.

Following the notation of Amigo Deliverable D2.1 [2] we use the term capability as a high-level functionality provided or required by the specific component. Furthermore we distinguish these capabilities into required (input) and provided (output) capabilities.

The capabilities of *IdialogueManagement* are:

- *setDialogue/getDialogue:* Provided capability for an application/composite service to define a required Dialogue, including resolving of conflicts.
- *notifyUser:* Provided capability for presenting all system notifications/messages to the user.
- *askUser:* Provided capability for engaging a dialog with the user in order to gather specific information (i.e. what the user wants to do with respect to a certain device or situation)
- *identifyUser:* Provided capability for user identification
- *verifyUser:* Provided capability for user verification
- *getFeedback:* Provided capability for an application/user to get feedback on which dialogues are recognized (for user calibration via *adaptRecognition*)
- *adaptRecognition:* Provided capability for training and adaptation of recognition process (modality specific adaptation of recognised patterns as well as their semantics).

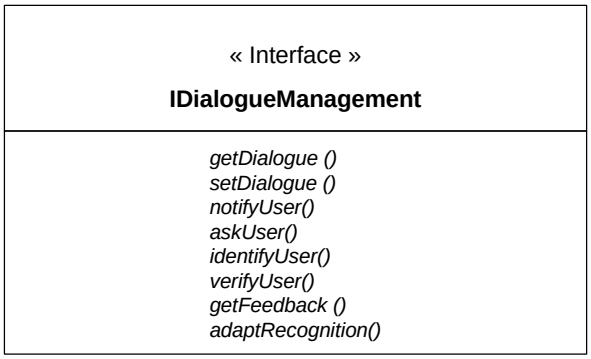


Figure 5.6: IUserManagement interface

The *IRequestHandler* and the *IDirectManipulation* are the interfaces to the User Modeling and Profiling Service.

IRequestHandler

- *getSetting:*
  - Required capability for getting user model and modifying accordingly the interaction strategy (i.e. favorite sequence of actions)
  - Required capability for getting personalized UI related settings from the user profile (e.g., voice stamp, vocabulary (different user can use different words))

IDirectManipulation

- *updateUserProfile:* Provided capability for learning user preferences and updating their profile (i.e., update of voice stamp)
- *getUserID:* Required capability for getting user ID after a successful user recognition

The *IAccessContextInfo is the* interface to the Context Manager Service.

IAccessContextInfo

- *getContextInfo:* Required capability for getting information from context aware service with respect to presence of people and devices.
- *updateContextInfo:* Provided capability for sending gathered knowledge about the context of the users and the system itself.

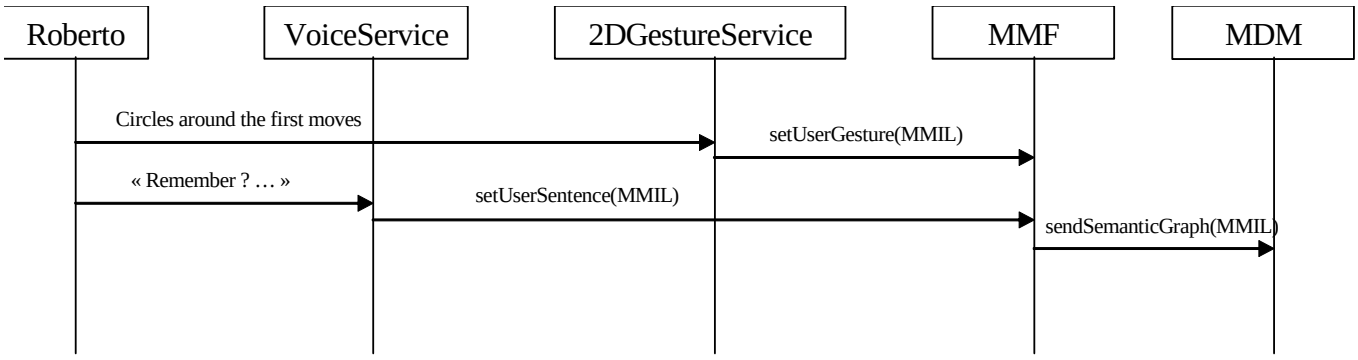
Base Middleware

The User Interface Service will use service-oriented mechanisms to discover and register to microphones, speakers, displays and other UI devices from the Base Middleware.

5.3.7 Dynamic behavior

Example*: Roberto is playing a board game with Peter. They have just finished their game, and comment the moves they have done on the screen. Roberto shows the first moves with his electronic pen: "Remember? You already defended like that some time ago. You had problems then to gain the center."*

The system blinks a small message "game, Roberto/Peter, 5th April 2004" on the screen. Roberto clicks on it: the old moves are displayed on the screen.

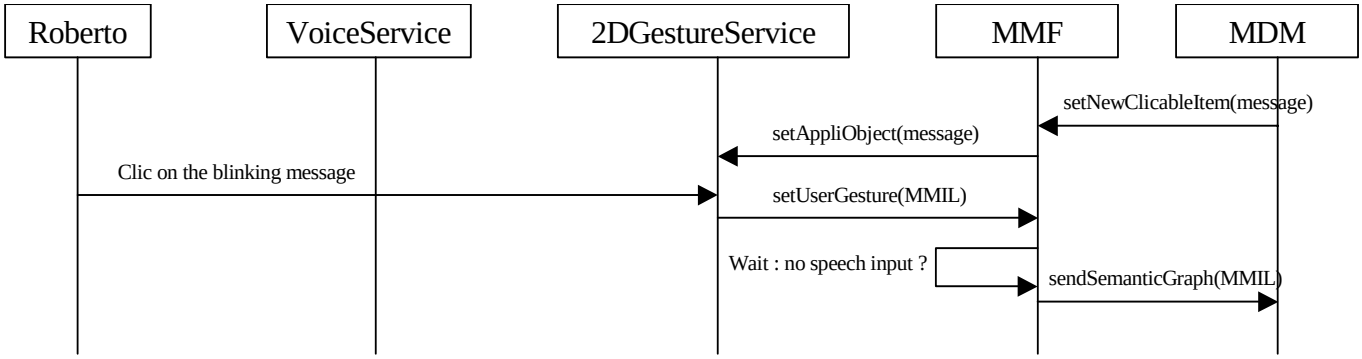


The semantic graph received by the MDM contains a number of events that describe Roberto's sentences, with every references and designation gestures solved, e.g.:

Defend(Peter,moves,…)

HaveProblems(Peter,game#1024,…)

The MDM and the chess application react to this semantic content, and prints out the blinking message. It also updates accordingly the application objects in the MMF internal model.



The semantic graph received by the MDM contains an event

select(message)

The MDM transfers this event to the application, which removes the message, displays the old game, and updates accordingly the application objects in the MDM, MMF and Gesture Service.

6 Privacy and Personal Security Mechanisms

6.1 Introduction

The role of the Privacy and Personal Security Mechanisms is to ensure personal privacy and security for the highly personalized Amigo system. The major challenge is to account for the implications induced by acquiring, collecting and inferring personal information of users in a social and perceptual acceptable and ethical responsible way. Intelligent User Services require the tracking and collection of significant portions of users' everyday activities and interactions to compose user profiles and to model the context in which these user behavior's and interactions occur. The disclosure of this private information to other parties, whether it be friends or family, service providers or commercial traders, in return for benefits like receiving a desired personalized and context-aware service or specific activities being taken care off, induces a delicate balance that needs to be maintained (Amigo Deliverable D1.2 [2]) and protected.

Intelligent User Services rely for their operation on appropriate and sufficient information of the users. This information may include their identity, their usage patterns of systems and services, preferences and dislikes, and actual time and dates of usage. Information might be collected by explicit means, i.e., input is consciously given by the user. Or by implicit means, i.e., input is acquired without explicit intervention of the user. This might imply that the system continuously records sensitive and private information about users and makes inferences and predictions about their behavior based on personal information while they are unaware of this and have no control. Crucial then for Intelligent User Services is building and maintaining the trust relation with their users. Creating trust in these services relies upon creating a perception of balance between costs and benefits for the users. The potential privacy risks for people are great and little is known about the actual perception of users when they are exposed to such context-aware environments.

Designing for privacy and personal security/safety right from the beginning is a basic concern of the Amigo project. As a consequence, mechanisms for privacy and security will be explored and in first instance produce guidelines for developing and integrating privacy enhancing features in the different Intelligent User Services.

At issue are, the implications of employing sensing technology for recording and monitoring the behavior of people, and storing and using these data for automatically selecting and adapting functionality and services to users. In addition, people provide, detailed personal data and behavioral information in either explicit or implicit fashion to generate profiles, identify preferences, and model familiar habits. These implications are difficult to comprehend by users and designers alike. They are insufficiently reflected in the user requirements and only expressed in a very generic fashion. This implies that in order to operationalize privacy and security requirements, an intermediate phase is necessary in which guidelines that focus on the specifics of the Amigo services are addressed.

In other words, Privacy and Personal Security is not like the other Intelligent User Services of the Amigo project, but a requirement that needs to be fulfilled by all. This implies that Privacy and Personal Security will be treated as a mechanism that is orthogonal to the other Amigo Intelligent User Services. Figure 1.2 (page 10) reflects this vision.

Privacy and personal security issues play a role in nearly all elements of the Amigo scenario. But, in particular, the following topics that may affect the perception of privacy and personal security by users can be distinguished:

- Acquisition of surveillance data of users.
- Using implicit means for collecting information.

- Disclosing personal information to other people and services, for example, personal preferences, habits, lifestyles, health information, location information, and social availability.
- Storage risks, i.e., private information might get lost, hidden or stolen if it is stored locally, or stolen, attacked, or misused if it is stored on a server or a network of servers.
- Dependability risks, i.e., loss of convenience when the system is not available, or malfunctioning of appliances in manipulated home systems.
- Affecting social relationships, for example, relations between parents and children, spouses, friends, social groups.

6.2 Privacy and Personal Security in the Amigo Scenario

The generic user requirements for the Amigo system can be summarized as follows. The system should provide concurrently the appropriate information to the right persons for the appropriate occasion at different locations and at the right time. It should provide controllable access and respect individual preferences and authorities. The Amigo system should be a multi-user system that can be personalized and customized, such that it:

- Generates and maintains privacy profiles,
- Enables user authentication,
- Tracks user behavior,
- Detects physical intrusions,
- Assesses situation conditions, and
- Is context-aware.

This implies that personalized mechanisms must be designed to exploit the benefits of personalization to the maximum while protecting user privacy. In other words, it is crucial to keep the user in control of which data will be collected and how they will be used.

A further aspect of such mechanisms is to provide personalized privacy support in group-situations, especially when accessing personal information on “non-personal displays” in common areas, like the kitchen, living room, or hallway. While most developers rely on social protocols or do not address privacy questions at all, there are very few approaches to actively support privacy on large public displays (Roe05 [20]). In most cases, e.g. (Rek98 [21]), (Gre99 [22]), additional private displays are used to generate and present personal information, while public information is displayed on a shared large display. A different approach using a stereographic display and special shutter glasses is described in (Sho01 [23]). Personal privacy is maintained through the filtering of the information on the shared display. Users wearing shutter glasses will see the public information as well as their own private information, while other person’s private data is not visible to them. Although these systems support individual privacy in an adequate way, they always require additional personal devices, like PDA’s or shutter glasses. Since public displays in home environments are mostly used for “walk-up-and-use” applications, like for example spontaneous video communications with remote family members, the existing solutions are not very suitable.

The development of guidelines for the Privacy and Personal Security Mechanism will commence by focusing on elements from the Amigo scenarios. Additions to these scenarios will be made if deemed appropriate.

6.2.1 Amigo Scenario Examples

1. Usher

Scene 2, elements 9, 10, 11, and 12 of the Amigo scenario.

> Amigo functions as an usher-service for Maria and Jerry. This service recognizes people at the front and patio doors of the house and opens the door(s) for them if Maria and Jerry have authorized them. (*The usher-service has to ask authorization to the security & privacy component for people at door which previously have been authenticated. The security & privacy component has to register the information (who and when) related to the authorized people.*)
>
> Amigo, like a real professional usher can notify the inhabitants, can take visiting cards *(Extra information about visitors can be retrieved and stored.)*
>
> and show personalized marks of the people that are present in the house, for example, a picture, a note, light or a color code. *(The information stored by the usher about who is at home or about who has come to home is only accessible by authorized users.)*
>
> Amigo can take over housekeeping tasks, starting and stopping the working of appliances at a desired time with the correct settings for duration, dosing, and temperature. It can even detect the presence of inappropriate objects in the appliances. *(How the appliances are used can be stored for any maintenance service.)*
>
> Amigo as kitchen chef downloads recipes and cooking programs to the kitchen and displays them for easy food preparation, i.e., cooking along with the video. Moreover, the recipes always take the status of the provisions in the kitchen into account. *(How the appliances are used can be stored for any maintenance service.)*
>
> Amigo maintains the overview of the food and household stock and generates shopping lists at predetermined time intervals. The shopping lists are personalized, but they take items that are on special offer, seasonal variations and nutritional balance into account. *(Shopping lists or any shopping behavior of the user can be stored.)*

2. Virtual Identity

Virtual Identity is a scenario element derived from Scene 2, element 9 and Scene 1, element 5.

> Roberto comes home from work and changes his virtual identity from “professional” to “private”. (*Every user should have different virtual identities allowing him to uses different profiles in different contexts*.)
>
> He uses his personal mobile device to change his virtual identity. The change of his virtual identity is done via a very intuitive and easy interaction. The system changes the level of privacy protection according to the default setting of the new identity. (*There should be an easy way to switch between different profiles.*)
>
> Roberto quickly checks who is logged in his awareness system and informed about his presence and availability. (*The user should always be aware of the information that is currently collected and know to whom it is accessible.*)
>
> He uses his personal mobile device to temporally overrule the privacy settings. The system adapts the privacy settings and mediated awareness information for the new virtual identity. (*The user should always be in control. For ease of use user-defined pre-setting can be used, but it should always be possible to easily overrule the setting*.)

3. Sharing & Caring

This scenario example expands on Scene 3 elements 13 and14 and illuminates some of the consequences with regard to Privacy and Personal Security.

After the movie, Maria decides she wants to talk to her father. When Maria walks towards one of the screens the image of John in his own house zooms in and the two talk about what they did during the day and their plans for the coming night. This privileged mode of communication between John and Maria is automatically suspended when Madeleine comes in. (*The system should be aware of the people present and adapt the way of information representation to the current context.*)

John has set her preferences to a semi-privacy mode in this situation, and the image presented to his daughter is partially obscured through a virtual Venetian blind. *(When using large public displays the system should hide private information from other users according to personal privacy profiles.*)

Maria can no longer make out their conversation, but her father's presence is still indicated. She says 'bye, will talk to you later'. (*The information should be concealed in an appropriate way, showing as much information as possible, but hiding the data that is defined as private by the user.*)

6.2.2 Related User Requirements

The following most relevant user requirements from the list in Amigo Deliverable D2.1 [3], Chapter 9 (Appendix A in this document) are addressed in the scenario examples for Privacy and Personal Security:

1: Easy and intuitive interface ("direct" communication, natural and easy interaction, easy to change preferences. The user must not need expert knowledge or assistance from a service provider.

2: Not being used for surveillance. Also people don't want to collect too much data and make those available online.

3: Enable individual settings and preferences.

4: Be configurable by the user or service provider.

5: Be movable, in case of moving house.

12: The system must be secure, safe and protect the privacy of all users. Personal data and preferences must be protected not only against intruders, but also against other users.

26: The system should protect against abuse, intrusions, loss of data, and house hackers.

27: The system should provide controllable access and respect individual preferences and authorities.

36: The system should support feeling of connectedness to family and friends.

37: The system should support 'trusted' relationships, e.g. meeting new people mainly through mutual friends.

6.3 Conceptual Data Model

Though privacy and security is in itself difficult to encase in a model, there are some concepts that are needed for a system to honor privacy. These concepts are, at least, *PrivacyGroup, PrivacyRules and PrivacyFilter.* These concepts will be further detailed in Amigo Deliverable D4.2 [7]. Here a global identification and description is provided.

PrivacyGroup can be defined as a collection of users, centered around one user, typically all combinations of all users can compose a privacy group, including 'none' and 'all'.

Typical examples of a *PrivacyGroup* would be 'personal' (only the user), 'intimate' (e.g., spouses), 'family' (i.e., defined as: *i*) all the members of a household under one roof; *ii*) a group of persons sharing a common ancestry (AMER76 [24])), and 'friend' (i.e., defined as: a person whom one knows, likes and trusts [24]), etc. Situations that are not explicitly related to users (e.g., context logging while no-one is in the house) are considered 'neutral', which is

also a *PrivacyGroup* (no users). That is, one of the properties of *PrivacyGroup* is 'composition', which can have the values: 'neutral', 'personal', 'family', and 'friend'.

Depending on the data a service processes, it needs to use *PrivacyGroup* in specific ways, for example:

- From information acquired from *ContextSource* the Context Management Service, *PrivacyGroup* can determine when the location of users is known and where this location is (this implies that location management is a requirement for supporting privacy in an adequate way). As a rule, all context information that is gathered for a certain *PrivacyGroup*, defined per location for the users that are present at that location, is only valid for these groups of users.
- From information acquired from *ContentAccess* of the Content Storage and Distribution module in the Base Middleware, *PrivacyGroup* can determine ownership of content. In this case, the notation of 'owner' is mapped onto the central entity, which defines access to a certain group or class of content. This enables the concept of group access that is not unlike the one used within UNIX. When several users are, for example, the owner of one file, then access is derived by group intersection (AND operation). Again, reasoning on content type, genre, etc. can help to connect certain content to certain *PrivacyGroup*.
- From the User Modeling and Profiling Service, user profiles can be acquired. A user profile is considered personal. Users might decide that parts of their user profile belong to different *PrivacyGroups*. Based on context information obtained from *ContextSource* of the Context Management Service, *UserProfile* can belong to a related *PrivacyGroup*,

In addition to end-users, other stakeholders also have privacy and security requirements. For example, when certain documentation or content that is owned by a company is accessed by a person, who has the rights to do this, and who shares this content with other users, who have not been granted access or have been identified by the company, privacy rules are infringed. This implies that 'stakeholder' is a property of *PrivacyGroup* and that is value determines whether services will be contained in some *PrivacyGroup*s and excluded from others.

PrivacyRules can be defined as the rules that the services and applications have to defer to in order to warrant the privacy conditions of the users. Aggregated reasoning on *PrivacyGroups* will be conducted to prevent that services, which might not always have sufficient information about users and context, infringe on the user's privacy. 'Ground rules' are provided for this. It is, however, the users who control their own privacy and decide on hiding and sharing information. This will be modeled in *PrivacyRules*.

The syntax for *PrivacyRules* resembles the descriptions in the *PrivacyAndSecurityMonitor* of the Awareness and Notification Service In this way, 'Privacy Awareness' can be seen as a special kind of awareness (but a very important one).

Typically, these rules will allow or deny certain actions to services and services need to check the rules in the rule database. Description of a rule will often be related to *ContextInfo, PrivacyGroup, UserProfile* or other content. Several aspects of the Amigo services gather in a *PrivacyRule*.

Amigo aware services and applications need to defer to all *PrivacyRules*. Authentication and authorization as provided by the Amigo Base Middleware cannot ensure personal privacy. It can protect personal data and in this way facilitate the protection of privacy and personal security. The most important rule has a different character: Services that infringe on the privacy rules will not be used.

Finally, the functionality that allows a service to check an action it intends to execute against the set of *PrivacyRules* can be defined as a *PrivacyFilter*.

In order to further develop the conceptual model, knowledge is needed to specify the privacy rules and the conditions for the filter functionality. This knowledge will be acquired from privacy guidelines that are derived from empirical investigations of situations described in the Amigo scenario.

6.4 Privacy Guidelines

Adequate privacy protection is a widely discussed topic since the early days of ubiquitous computing (Wei91 [25]), (Bel93 [26]) and several authors (Bel01 [27]), (Lan01 [28]), (Lah05 [29]) developed guidelines for privacy enhancement in pervasive computing environments. These guidelines provide a first step for building awareness and creating a theoretical framework, they lack, however, sufficient specification to be deployable by engineers and developers of services and applications. Since the Amigo system takes a user-centered approach, part of the scenario assessment work done in WP1 can be considered as already taking some of these guidelines into account. These results showed that recording, in whatever fashion, seemingly innocuous data about daily activities, induce complex interrelated effects on people's perception of privacy and security. The scenario elements that have been identified in 6.2 will be used to disentangle these relationships and generate guidelines for the Amigo applications and services. The situations that are of major importance and that are hidden within the scenario elements concern:

- Acquisition of surveillance data of users
- Using implicit means for collecting information
- Disclosing personal information to other people and services
    - Personal preferences, habits, lifestyles, health information, location information, social availability.
- Storage risks
    - Locally: private information might get lost, hidden or stolen
    - Server or network: attacked or misused
- Dependability risks
    - Loss of convenience when the system is not available, or malfunctioning of appliances
- Affecting social relationships
    - Relations between parents and children, spouses, friends, social groups.

6.4.1 Approach to generate guidelines for Privacy and Security

The focus of the Amigo system is on the networked home environment in which people conduct sequential activities and operations. N.B. this is not about one end-user conducting one task with one device. Different configurations of user tasks and activities are possible and series of interrelated processes occur. Monitoring these user activities with various sensors and other means of recording provides an accumulation of data that carry an explosive load of intimate personal information.

To generate guidelines for privacy and security for the Amigo services and applications, the following activities are foreseen. First, starting from an Amigo scenario element an analysis needs to be made of the user tasks and user activities in this particular scenario. Different cases have to be distinguished, for example, one and multiple persons, different relations between persons, different sequential configurations of tasks (people can do their things in different orders), and different means for data collection (location of sensors, other monitoring and recording devices). The possible variation in cases is partly determined by the scenario

description and partly by assumptions because the scenario descriptions can't foresee all complications. For each of these cases privacy and security aspects should be identified and elaborated on. These activities will provide sets of explicit privacy and security aspects and their corresponding conditions. Second, these sets can be used to design experimental environments with set-ups in which user tests can be conducted to generate implicit and unforeseen privacy and security aspects and to assess the consequences of perceived threads to privacy and security. Consequently, the results of these tests can be translated into dedicated guidelines for the design of Amigo services and applications.

In short, the objectives of this empirical approach are to design, prototype and evaluate different realistic configurations of Amigo scenario elements with regard to privacy and obtrusiveness factors and to generate guidelines for perceived privacy and security that can be used for system design and implementation.

The following activities will be conducted:

- Starting from an Amigo scenario element an analysis user tasks and activities are specified. Different cases will be identified, for example, one and multiple persons, different social relations between people, different sequences of activities, different means for information collection (sensors, recording devices, etc.), different means for awareness notification. Subsequently, privacy and security aspects for a selected set of cases are identified and transformed in research questions that can be empirically addressed.
- Design and prototype experimental settings in which the above privacy and security aspects can be investigated with users. This part of the task will exploit the available infrastructure, e.g., the Philips HomeLab, and other technologies that are being investigated in the Amigo project.
- Evaluate the prototype, design and conduct the experiments with users under the appropriate conditions.
- Analyze the results and derive guidelines for Amigo system design, such that they can be implemented as rule-based components in the Amigo Service Framework Architecture.

The results of this work will be used to support design decisions for the Amigo architecture that are grounded on empirical evidence.

7 Conclusion

The Intelligent User Services and their interfaces that will be further developed in the Amigo project were presented. These Intelligent User Services are the building blocks for the Amigo applications and services. For each Intelligent User Service, the global concepts, modules and interfaces were explained, together with some examples of their dynamic behavior. These services are Context Management Service, User Modeling and Profiling Service, Awareness and Notification Service, and User Interface Service. A first outline was given of the privacy and personal security issues that are deeply embedded in the Amigo services.

The Intelligent User Services being developed in the Amigo project are grounded in the user requirements that were acquired in WP1. They match also with the proposal of a “people-oriented, empowering smartness” (Stre05 [30]) for designing smart environments. In contrast to more or less automatically acting home automation and control systems, this approach emphasizes to keep users “in the loop” and still in control by making decisions based on suggestions derived from information that is, e.g., collected by sensors in the home. The Amigo scenario functions as the binding factor between the Intelligent User Services and as a means of communication to explain and conduct walkthroughs to evaluate the architectural decisions. The Amigo project, in contrast to other projects in this domain, for example the ePerSpace project (ePe04 [31]), focuses on the Intelligent User Services *within* the home environment and not on bringing services to the home, i.e., the Intelligent User Services are building blocks in the Amigo architecture; they run on Amigo device(s) within the home. This implies that for Amigo a service is not defined as a product that is offered by a service provider (producer) to a client (consumer) with the aim of the service provider to obtain revenues, but as a set of coherent functionality offered by one Intelligent User Service to another service, application or end-user.

The Intelligent User Services exploit the capabilities of the underlying infrastructure that was developed in WP2 (System Architecture) and WP3 (Open Middleware for the Networked Home). For example, ontology descriptions are used for composition and definition, and the mechanisms provided by the open middleware are used by the Intelligent User Services for service discovery, late binding and interaction, for content distribution and for privacy and security to warrant features like single sign-on and integrated authorization. The complexity of defining the dependencies and interfaces should not be underestimated. Several factors contribute to this complexity.

First, the range of information sources that are available inside the home, between homes and outside in public spaces need to be exploited. A combinatorial explosion of possibilities is created as we are dealing with intricacies, such as, having multiple users, multiple devices, multiple physical contexts, multiple social arrangements. The Amigo scenario proved to be very helpful in reducing this complexity. The Intelligent User Services unevenly selected the 15 elements that compose the Amigo scenario. That is, if we consider the choices of the services as a vote, the elements 1, 3, 5, 9, 13 and 14 were addressed by at least 3 of the services. Scene 3, element 13 and Scene 2, element 9 were addressed by al services (including the Privacy and Personal Security Mechanism). These scenario elements, i.e., ‘Usher’ and ‘Caring & Sharing’, come closest to the needs of the services to research and develop as applications, to the user requirements, and to interface and integrate between the services. The Usher, selected as a scenario element to work on by the Intelligent User Services, implies the monitoring of all traffic in and out of the home of people and devices, knowing where people and devices are in the home and what their context and privacy conditions are, and assigning the appropriate actions with regard to content, form and function of notification. The Caring&Sharing, also selected as a scenario element to work on by the Intelligent User Services, especially to support the extended home environment, implies the monitoring of activities of people between homes to support caring for each other and sharing daily activities and interests. It is assumed that the activities in WP5, Home Care and Safety, WP6, Home Information and Entertainment, and WP7, Extended Home Environment will use the Usher and

Caring & Sharing as starting points for the development of the demonstrators. Note, that these two top-ranked scenario elements are not excluding the other parts of the Amigo scenario. The Intelligent User Services will also use the other scenario elements.

Second, the project structure follows a 'top-down' approach. That is, starting from user requirements, the system architecture is being designed and specified. This approach is fully justified and correct. But, in the case of a complex project like Amigo, addressing a very complex architectural problem in a novel domain, it is very difficult for users and architects alike to perceive what it is all about. This is an epistemological issue: people first need concrete examples before they can start thinking at higher levels of abstraction (Sim69 [32]), (Pia73 [33]). The Amigo project team is in the midst of this process. Focusing on the scenarios and using 'thought-experiments' proved to be a productive approach in defining the Intelligent User Services, communicating about them, and getting the global descriptions worked out.

Third, conventions are crucial for a project like Amigo, in which different partners and different disciplines come together. This is part of the design process and needs several iterations. A first start was made in preparing this deliverable. But additions and more specifications are foreseen for the follow-up activities in the project.

Fourth, the Amigo system has an open architecture, is distributed and operates in real-time. That is, the amigo architecture concerns system architecture and process architecture. These requirements are being addressed in WP3, but how the Intelligent User Services fit into this needs to be worked out. For example, what happens when one of these building blocks fails and top priority user requirement of 'it should always work' is being violated? Simplifying the scope by focusing on a few scenario elements by all Intelligent User Services, such that the results can be exploited in demonstrators (WP5, WP6, WP7) provides the possibilities for exploring fail-proof and foul-proof conditions.

Fifth, the scope of the Amigo project is divers and complex. For example, we have to address specific questions like: How to handle multiple profiles and dynamic modeling of behavioral patterns? How does an intelligent user interface dialogue manager handle multiple applications? How to enable users to input multiple rules? May controllers act as event sources for other controllers from other domains? Such questions are already complex to address within a single Intelligent User Service, they become more complex when the dependencies between the different services need to be taken into account.

The next deliverable (Amigo Deliverable D4.2 [7]) of the Amigo project will detail the architectural relations between the Intelligent User Services, the subcomponents within the Intelligent User Services, and the operations of these services within the Amigo service framework. In this document the first onset is given for the Intelligent User Services architecture.

References

[1] Amigo D1.1, 04 Amigo Deliverable D1.1: “Amigo User Research Results”, S. Un (ed.), IST-004182 Amigo, January 2005.

[2] Amigo D1.2, 05 Amigo Deliverable D1.2: “Report on User Requirements”, M. D. Janse (ed.), IST-004182 Amigo, April 2005

[3] Amigo D2.1, 05 Amigo Deliverable D2.1: “Specification of the Amigo Abstract Middleware”, N. Georgantas (ed.), IST-004182 Amigo, April 2005.

[4] Amigo D2.3, 05 Amigo Deliverable D2.3: “Specification of the Amigo Abstract System Architecture”, M.D. Janse (ed.), IST-004182 Amigo, June 2005.

[5] Str05 Streitz, N.A., Nixon, P. (2005): The Disappearing Computer. Guest Editors'. Introduction to Special Issue of Communications of the ACM, Vol. 48 March, 2005. pp. 33-35.

[6] Amigo D3.1, 05 Amigo Deliverable D3.1: “Detailed design of the Amigo Base Middleware infrastructure”, N. Georgantas (ed.), IST-004182 Amigo, September 2005.

[7] Amigo D4.2 Amigo Deliverable D4.2: “Report on detailed intelligent user services design”, B. Kladis (ed.), IST-004182 Amigo, in progress.

[8] Szy98 Szyperski, C. (1998) *Component Software - Beyond Object-Oriented Programming*, Addison-Wesley / ACM Press, 1998.

[9] May05 Mayrhofer, R. (2005): “Context Prediction based on Context Histories: Expected Benefits, Issues and Current State-of-the-Art”. In Proceedings of the 1st International Workshop on Exploiting Context Histories in Smart Environments (ECHISE’05), held in conjunction with PERVASIVE’05. 11 May, 2005, Munich, Germany.

[10] Pra05 Prante, T., Petrovic, K., Stenzel, R. (2005): ContextDrive – Personal Information Management for Ubiquitous Computing (invited paper). i-com magazine, 1/2005, 2005. pp. 41-47.

[11] Awa04 AWARENESS Service Infrastructure: “Architectural Specification of the service infrastructure”, P. Dockhorn (ed.), Freeband AWARENESS project, http://awareness.freeband.nl, Dec. 2004.

[12] Gem04 GEMINI Deliverable D3.2: “Final Prototype of the Application Generation Platform (FP-AGP)”, Technical Report, GEMINI (IST-2001-32343), June 2004.

[13] Ric98 Rich, E., “User modeling via stereotypes”. In Maybury M. and Wahlster W (eds.): Readings in Intelligent User Interfaces, Berlin: Springer-Verlag, 1998.

[14] Hin03 Hinze, A. M. (2003), “A-MEDIAS: Concept and Design of an Adaptive Integrating Event Notification Service”, PhD thesis, Freie Universität Berlin, 2003

[15] Doc05 Dockhorn Costa, P., Ferreira Pires, L. & van Sinderen, M. (2005), “Architectural Patterns for Context-Aware Services Platforms” In: Proceedings of the International Workshop on Ubiquitous Computing (IWUC 2005) in conjunction with the 7th International Conference on Enterprise Information Systems (ICEIS 2005), 24-25 May, 2005, Miami, USA.

[16] Ipi01 Ipiña, D. L. de and E. Katsiri (2001), "An ECA Rule-Matching Service for Simpler Development of Reactive Applications", Published as a supplement to the Proceedings of Middleware 2001 at IEEE Distributed

Systems Online, Vol. 2, No. 7, November 2001

[17] Lan04 F. Landragin, A. Denis, A. Ricci and L. Romary. Multimodal Meaning Representation for Generic Dialogue Systems Architectures. In Proc. LREC 2004.

[18] Kel05 Kela, J., Korpipää, P., Mäntyjärvi, J., Kallio, S., Savino, G., Jozzo, L., Di Marca, S. Accelerometer-based gesture control for a design environment. To appear in Personal and Ubiquitous Computing special issue on Multimodal Interaction with Mobile and Wearable Devices, Springer-Verlag 2005. In Press.

[19] Mag04 Magerkurth,C., T. Engelke, M. Memisoglu: Augmenting the Virtual Domain with Physical and Social Elements. In: 1. International Conference on Advancements in Computer Entertainment Technology (ACM ACE 2004), Singapore, ACM Press, June 3-5, 2004. pp. 163-172.

[20] Roe05 Roecker, C. (2005): Providing Personalized Privacy Support in Public Places. In: Proceedings of the third Annual Conference on Privacy, Security and Trust (PST 2005), pp. 217 - 220.

[21] Rek98 Rekimoto, J. (1998), "A Multiple Device Approach for Supporting Whiteboard-based Interactions". In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 1998), pp. 18 – 23.

[22] Gre99 Greenberg, S., Boyle, M., LaBerge, J. (1999), "PDAs and Shared Public Displays: Making Personal Information Public, and Public Information Personal". Personal Technologies, 3, 1, pp. 54 – 64.

[23] Sho01 Shoemaker, G.B.D, Inkpen, K.M. (2001), "Single Display Privacyware: Augmenting public displays with private information". In: Proc. of the ACM Conference on Human Factors in Computing Systems (CHI 2001), pp. 522 – 529.

[24] AME76 "The American Heritage Dictionary", Houghton Miffin Company, Boston USA, 1976.

[25] Wei91 Weiser, M. (1991), "The Computer for the 21st Century", Scientific American, 265, No. 3, pp. 94 – 104.

[26] Bel93 Bellotti, V., Sellen, A. (1993), " Design for Privacy in Ubiquitous Computing Environments". In: Proceedings of the 3rd European Conference on Computer Supported Cooperative Work, (ECSCW 93), G. de Michelis, C. Simone and K. Schmidt (Eds.), Kluwer, 1993, pp.77–92.

[27] Bel01 Bellotti, V., Edwards, K. (2001), " Intelligibility and Accountability: Human Considerations in Context Aware Systems". HCI: Special Issue on Context-Aware Computing, Vol. 16, pp. 193-212.

[28] Lan01 Langheinrich, M. (2001), "Privacy by Design - Principles of Privacy Aware Ubiquitous Systems". In: Proceedings of Ubicomp 2001, volume 2201 of LNCS, Sep. 30- Oct. 2, 2001, Atlanta, GA., pp. 273-291.

[29] Lah05 Lahlou, S., M. Langheinrich, C. Röcker (2005), "Privacy and Trust Issues with Invisible Computers". In: Communications of the ACM, Vol. 48 March, pp. 59 – 60.

[30] Str05 Streitz, N., C. Röcker, T. Prante, D. van Alphen, R. Stenzel, C. Magerkurth (2005), Designing Smart Artifacts for Smart Environments. IEEE Computer, March 2005, pp.41-49.

[31] ePe04 ePerSpace project (2004), Deliverable D1.1, Description of Scenarios and Specifications. IST project No. 506775. http://www.ist-eperspace.org/

[32] Sim69 Simon, H. A. (1969), “Sciences of the Artificial”. Cambridge, Mass.: MIT Press.

[33] Pia73 Piaget, J. (1973), ”To Understand is to Invent”. Harmondsworth, Middlesex, England, Penguin Books Ltd. 1973.

Acronyms

3GPP: 3rd Generation Partnership Project

CE: Consumer Electronics

CORBA: Common Object Request Broker Architecture

CPS: Core Profile Service

DNS: Domain Name Server

ECA: Event Condition Action

ENS: Event Notification Service

EPS: Expanded Profile Service

GPS: Global Positioning System

GUI: Graphical User Interface

IETF: Internet Engineering Task Force

IMS: Instant Messaging System

IOP: Interoperability Protocol

MAC: Medium Access

MDM: Multimodal Dialogue Manager

MFM: Multimodal Fusion Module

MMIL: MultiModal Interface Language

OWL: Web Ontology Language

PIDF: Presence Information Data Format

QoC: Quality of Context

RFC: Request for Comments

RFID: Radio Frequency Identification

RPID: Rich Presence Information Data

SIP: Session Initiation Protocol

SOAP: Simple Object Access Protocol

SOTA: State of the Art

UMPS: User Modeling and Profile Service

WiFi: Wireless Fidelity

WP: Work Package

WP: Workpackage

XML: eXtensible Markup Language

Appendix A

Prioritized user requirements from high (0) to low (6) and numbered from 1 to 37 related to technical requirements from Amigo Deliverable D2.1 [2].

Priority	No.	User requirement	Technical Requirements
0	1	Be easy to use and to configure – no need for programming by the user	Device discovery
			Service discovery
			Service composition
			Re-configuration
			Assisted/ automatic configuration of services
0	2	Not being used for surveillance (from the users' perception)	Multi-user system
			Manual service initiation/ interaction
			Privacy profiles
			Security profiles and user authentication
0	3	Enable individual settings and preferences	Personalization and customization
			User tracking
			Context awareness
0	4	Be configurable by the user or service provider	Remote configuration & monitoring
			Re-configuration of network and devices
0	5	Be movable, in case of moving house	Customization
			Ad-hoc interoperable networking
			Service migration
			Re-configuration of network and devices
0	6	Be extensible - easy to upgrade	Componential
			Component based middleware
			Standardized services
			Automatic updates
			Service discovery
			Re-configuration of network and devices
			Billing server
			Device discovery

0	7	Be flexible	Ad-hoc interoperable network
			Context awareness
			Device discovery
			Service discovery
			Interoperable (domotic) interfaces
			Interoperable interfaces
0	9	Be modular	Componential
			Component based middleware
			Fault detection and recovery
			Device discovery
			Service discovery
			Interoperability at network, device, middleware and interface level
0	10	Be maintenance free (i.e., no need for maintenance by the user)	See 6
1	11	The user must always remain in control of the system and never the other way around	Have user confirmation; keep user in the loop
			Multi-user system
			Unobtrusive interfaces
			Privacy profiles
1	12	The system must be secure, safe and protect the privacy of all users	See 2
1	13	The system must provide an added value to existing systems for multiple users at the same time	Multiple-user system
			Group/ community profiles
			Personalization and customization
			Distributed system
			Multicasting
1	14	The system should never unnecessarily replace direct interaction between people	Unobtrusive interfaces
			Context awareness
1	15	The home comfort should always be maintained and not be subservient to the system	Context awareness
			Light sensors, kinetic sensors
			Sound adaptation
			Personalization and customization
			User tracking

2	16	The system should provide concurrently the appropriate information to the right persons for the appropriate occasion at different locations, i.e., filter information, provide resumes, according to user preferences (note people refer to existing services that they know)	Context awareness
			Multi-user system
			Situation assessment
			User tracking
			Privacy profiles
			Personalization and customization
2	17	The system should enable easy access and usage of information and data from different sources.	Data convergence mechanisms
			Standardized interfaces (API's)
			Ad-hoc interoperable networking
2	18	The system should support storage and archiving of data in diverse ways.	Personalization and customization
			Storage system
			Replication
			Multimedia communication
2	19	The system should support having control over data and information for best performance	Context awareness, classification of data
			Quality of service
			Situation assessment, ontology's
			Distributing data
3	20	The system should reduce the time needed for household chores and where possible do cleaning jobs	Management and overview of domestic services
			Domotic bus driver
			Domotic device drivers
3	21	The system should integrate and combine functionality of appliances	Device and service discovery
			Interoperability
3	22	The system should be energy saving	Power aware
			Prioritization of devices and services
			Context awareness
3	23	The system should be cost saving	Billing server
3	24	The system should maintain the appropriate environmental conditions of the house (temperature, humidity, light, air, dust, mites, etc.)	Context awareness
			Controllers, sensors, actuators
			Personalization and customization
			User tracking
4	25	The system should support the activity organization and planning for multiple persons at home, between homes and between home and work	Context awareness
			Automatic data collection and notification
			Multi-user system
			Sharing information between users

4	26	The system should protect against abuse, intrusions, loss of data, house hackers	Data replication
			Security profile and user authentication
			Detecting physical intrusion
4	27	The system should provide controllable access and respect individual preferences and authorities	Security and privacy profiles
			Multi-user system
			User authentication
4	28	The system should support alignment of individual and group planning, updates and notifications.	See 25
5	29	The system should take context/environment conditions into account and be aware at any time of the local situation.	Context awareness
			Sensors
5	30	The system should support the integration of playing computer games in family routine, and approved settings.	Parental control
			Automatic community information
			Sharing of practices in a community
5	31	The system should support playing games and entertainment with multiple people in the same room or networked environment.	Multi-User system
			Adaptation of environment, See 24
			Physical interaction
			Additional processing requirements for games
6	32	The system should take implicit social rules of behavior into account	Personalization and customization
			Sharing user information
			Multi-user system
6	33	The system should support increasing number of communication moments in multiple different contexts	Always available system
			Componential user interface infrastructure
6	34	The system should enable communication with multiple people at the same time, e.g. broadcasting, democratic group planning.	Multi-user system
			Overlay Network
6	35	The system should support keeping in touch with select group of friends, no need to always be connected as “me”-time is just as important.	Service availability
			Transparent interfaces
			Context awareness
6	36	The system should support feeling of connectedness to family and friends	See 35
			Location management
6	37	The system should support ‘trusted’ relationships, e.g. meeting new people mainly through mutual friends.	Trust management
			Natural interfaces