

# AN INTELLIGENT USER SERVICE ARCHITECTURE FOR NETWORKED HOME ENVIRONMENTS

Carsten Magerkurth<sup>(1)</sup>, Richard Etter<sup>(1)</sup>, Maddy Janse<sup>(2)</sup>, Juha Kela<sup>(3)</sup>, Otilia Kocsis<sup>(4)</sup>, Fano Ramparany<sup>(5)</sup>

<sup>(1)</sup> Fraunhofer IPSI, AMBIENTE, Germany; <sup>(2)</sup> Philips Research, High Tech Campus, Nederland; <sup>(3)</sup> VTT Tech. Research Center of Finland; <sup>(4)</sup> LogicDIS, Patras, Greece; <sup>(5)</sup> France Telecom, RD-TECH-GRE, France

## ABSTRACT

This paper discusses a novel architecture of intelligent user services that address the domain of networked home environments. The work described in this paper represents the central achievements of the first 1.5 years in the work package on intelligent user services within the EU-IST funded research project Amigo. The discussed service architecture consists of five distinct services that address both user- and system oriented issues in Ambient Intelligence (AmI) home environments.

## INTRODUCTION

With the advent of mass-market enabling technologies that allow for interconnecting heterogeneous devices and equipping smart spaces with dedicated sensing infrastructures, the vision of smart homes that assist its inhabitants in their everyday tasks is becoming graspable.

The EU-IST funded research project “Amigo – Ambient Intelligence for the Networked Home Environment” addresses the issues of realizing such smart homes that do not only integrate various household appliances, sensors, and consumer electronics devices, but also aims at creating service infrastructures that use Ambient Intelligence methods to reason about the user’s goals and utilize context information in order to provide adapted functionality that acts appropriate in any given situation. The aim of Amigo hence is “to research and develop open, standardized, interoperable middleware and intelligent user services for the networked home environment, which offers users intuitive, personalized and unobtrusive interaction by providing seamless interoperability of services and applications” (Amigo Project, 1).

While lower level middleware that address issues such as interoperability of various devices, appliances, and protocols, service discovery, security, and scalability is an open and interesting research field in itself, this paper focuses on higher level services that build upon existing infrastructures and provide functionalities to

the users that have a perceivable effect on them. We define these services as “intelligent user services” in the sense that intelligence refers to adaptive capabilities with regard to being aware of the usage conditions, physical contexts, and social situations. Likewise, the notion of a user service refers to assisting users and taking the preferences and profiles of individual users into account in order to optimally adapt to individual needs. Lower level middleware related to AmI environments, e.g. for composing device services or related Quality-of-Service (QoS) issues are discussed in Vallée (16) and Papaioannou et al. (11) and are beyond the scope of this paper.

## DESIGN SPACE

The development of the service architecture presented here is addressed in the central work package “Intelligent User Services” that spans almost the entire 3.5 years project duration of Amigo. One of the most fundamental design alternatives that informed the initial planning and specification of the services is the nature of intelligence that we want to base our work on. The term “Ambient Intelligence” refers to information processing taking place in the periphery of our attention and hints at an implicit and largely automatic approach. However, as Streitz et al. (15) discuss in their analysis of smart homes and smart artifacts, there are ambivalent properties of implicit and automatic reasoning. They distinguish between two types of approaches: system-oriented, importunate and people-oriented, empowering ones. In the former approach, the services take certain actions based on autonomous reasoning such as adjusting the heating system or regulating lighting conditions based on assumed user goals. The problem here is that the user is not kept in control and incorrect decisions might have very negative consequences. Contrarily, people-oriented, empowering approaches focus on empowering users to make their own decisions and take mature and responsible actions. Keeping the human in the loop, however, might lead to information overload and might overwhelm users who do not want to deal with regulating services at all times.

Balancing the design goal of system-oriented autonomy and people-oriented control is an open research field. We believe that both approaches need to be combined in a flexible manner. Therefore, we have initially developed a set of intelligent user services that are integrating both approaches to varying degrees and allow applications that integrate them to set their individual foci. In the following sections we now discuss the initial set of services provided by the Amigo intelligent user service architecture. These services are currently in active development. Some are already deployed with initial functionality, others are currently in the specification phase. The first discussed service, the Context Management Service, provides the essential context information for Amigo aware applications.

## CONTEXT MANAGEMENT SERVICE (CMS)

Ambient intelligence (AmI) pushes forward a vision where our daily environment proactively serves our needs by understanding our activities, anticipating our needs and collaborating with us in achieving our daily tasks. To make this vision come true, the AmI environment needs to be aware of any context information that is helpful for identifying user's activities, needs and tasks. This information is to be found in the physical environment as well as from the users or from the computer systems or devices they use. We have adopted the following general definition of context from Dey (4): "*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*".

In the Amigo project, interactions between users and their physical environment is mediated by device based services. In a sketchy way, services are software components that hide physical devices and artifacts to humans, behind high level and user-friendly interfaces. Within this service oriented approach, context information affects the way services execute so as to optimally fit users' needs. Services can exploit context information in different ways.

Context information can (i) trigger a service so as to set up a pro-active environment, (ii) adapt the process of a service, or (iii) simply be presented to users using a variety of modalities in order to empower the smartness of users while they might interact with their environment. An awareness system displaying the status or availability of a person buddies is an example of this third type of context awareness.

Although any device endowed with processing and communication resources could potentially act as a source of context information, the Amigo project is

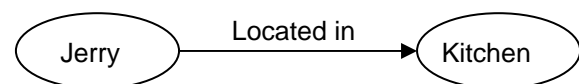
currently deploying a focused range of context sources that have been selected on the basis of the project target applications needs. Those context sources include:

- household appliances such as ovens, cookers, freezers, dishwasher and washing machine
- comfort sensors such as temperature, humidity, CO<sub>2</sub>, air cleanness sensors
- location sensors such as RfID, GPS, acoustic position estimation sensors
- user activities information provider such as a discussion topic recognizer
- components that aggregate basic context information such as a context history manager

In the Amigo project, information exchanged between those context sources and context aware services that exploit this information for adapting their execution is modeled using web services technology.

More precisely, we use the Resource Description Framework (RDF) for modeling context information. RDF is a language that has originally been designed for representing information about resources in the World Wide Web (RDF-CONCEPTS). Using RDF, resources are described as a collection of triplets consisting of (subject, predicate, object). Thus, in our modeling approach a piece of context information is a RDF fragment which relates entities (subjects of the triplets that are instances of concepts or classes from the context ontology) to other entities (objects of the triplet) or to literal values (strings, numerical values, etc.) using the predicate component of the triplet. The predicate should be defined in the context ontology as a relation linking the subject class to the object class.

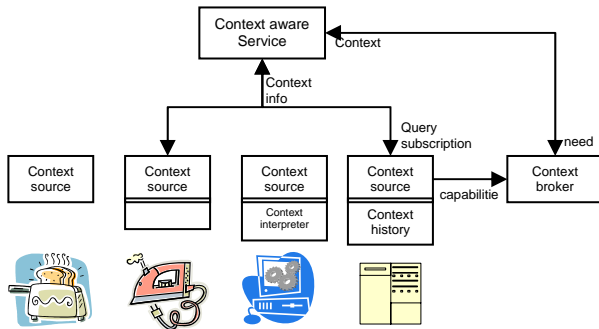
An example of context information about Jerry being located in the Kitchen" is represented graphically in pure RDF as:



While transmitting this information between context sources and context aware applications a textual serialized format based on XML is used. In RDF/XML notation, the same information will look like:

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-
rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://www.owl-
ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-
ontologies.com/unnamed.owl">
  <User rdf:ID="Jerry">
    <locatedIn>
      <Room rdf:ID="Kitchen"/>
    </locatedIn>
  </User>
</rdf:RDF>
```

Context sources and context aware services interactions for exchanging context information are organized within the following architecture (see figure 1):



**Figure 1 Context Management Components.**

The general principle behind this architecture is to provide context aware services, with a well-defined and unified interface to context sources, such as physical sensors. This interface adopts:

- RDF as the language to be used for exchanging context information among consumers and providers irrespective of the context's nature and the type of context sources
- A shared context Ontology to enable consumers and providers to understand each other
- RDQL as the language to be used by context consumer to query and subscribe to context information.
- Web service protocols as the mechanisms to be used by context consumers and context sources to connect and speak to each other

Context aware applications discover context sources of interest by applying their needs to context brokers. Context sources simply need to register to a context broker as soon as it is deployed in the environment.

To enrich this core functionality additional components have been introduced such as context interpreters and context history managers that enable to aggregate and reason on heterogeneous pieces of context information. Such reasoning is used for deriving high-level context information or analyze, recognize and predict temporal patterns of context events.

Compared to existing context management infrastructure (e.g. Salber et al (13), Khedr and Karmouch (9)) our approach promotes

- scalability through a distributed architecture
- dynamicity as context sources can appear and disappear in the environment with minimal impact on services currently running,
- functional interoperability through the adoption of the web service technologies
- semantic interoperability through the adoption of the semantic web technologies.

While CMS forms the backbone of both people-oriented and system-oriented approaches, the next discussed service is specifically geared towards people-oriented approaches by modeling users so that intelligent systems can utilize the context information from CMS and apply it to user profile information in order to realize highly adaptive applications.

## USER MODELING AND PROFILING (UMPS)

The User Modeling and Profiling Service (UMPS), presented in this section provides the methodology for context dependent personalization and adaptivity of applications and services in the Amigo environment. In a first step user profiles are built based on stereotypes and explicit user input, and in the second step these profiles are refined using the interaction/context history. User modeling, also known as personalization, user profiling, or adaptive user interfaces, is the key in the construction of interactive software systems which are able to recognize and to adjust themselves to the needs of particular users at every stage of use, whether these users be beginners or experts. Although user modeling has become a mature research field with demonstrable results, little progress has been made in the development of user modeling components for commercial software systems, mainly aimed at supporting personalization of e-commerce systems on the World Wide Web (Fink and Kobsa, 5).

User modeling and profiling has evolved from representation of groups of users using a certain system in certain conditions, to personalization of these systems towards individual users' preferences and requirements. In case of ambient intelligence systems personalization applies to different system components and application domains, particularly to user interfaces (e.g., graphical, gesture, voice-based) and content presentation (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 (Khedr and Karmouch (9), Rich (12)). 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.

## Service Description

UMPS provides the methodology to enhance the effectiveness and usability of applications, services and interfaces in the Amigo environment in order to:

- (a) tailor information presentation to user and context,
- (b) reason about user's future behaviour,

- (c) help the user to find relevant information,
- (d) adapt interface features to the user and the context in which it is used,
- (e) inform about interface features and information presentation features for their adaptation to a multi-user environment.

These goals are achieved by constructing, maintaining and exploiting user models and profiles, which are explicit representations of individual user preferences, personal data and system assumption on users' characteristics.

While the user modeling is certainly an important prerequisite to make the ambient intelligence user-friendly and accessible to the general public, it is also subject to potential dangers of misuse. To prevent misuse and resist to dangers of leaking user-related information in the Amigo system users will be made aware of the fact that the system contains a user modeling component, such that the user can decide whether or not to consent to being modeled by the system, or even to "switch off" the component or certain modeling subcomponents.

Personalization can be achieved either in completely autonomous mode, when the system decide what to present to the user based on the existing user profile data, or in a semi-automatic mode, when the system makes suggestions to the user, allowing him to feel in control and reducing the number of system mistakes. These two ways are not exclusive: in one context it is appropriate to act autonomously and in another context it is recommended to ask the user's opinion. However, at earlier stages of user-system interaction, when the confidence of the system on the validity of user profile data is low, the semi-autonomous mode should be preferred. UMPS, in a first step, will build a user profile based on the common sense knowledge of the system about its potential groups of users (stereotypes) and explicitly acquired data from the user. This first step makes possible to avoid the costly and elaborate construction of user ontology through the system itself, simplifying the initiation process of the adaptive system. In the second step, the user profile is refined based on the knowledge of events and facts of the Amigo system (interaction/context history, user's choice among system's suggestions, user's feedback). This refinement is a continuous process, involving and affecting all applications and services of the system that will make use, at runtime, of the profile data (i.e. the Awareness and Notification Service will make use of the user notification profile data).

## Architectural Design

UMPS has an add-on modular architecture (see figure 2), which is closely related to the implementation

phases. The basic inner shell of the architecture is the Core Profile Service. 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 Core Profile Service level, the update of the profile will be based on static modeling methodology. The major components of the Core Profile Service, providing the functionality necessary for the interfaces offered to other system services and to the applications domain, are:

(a) The Reasoning Module, which is responsible for exploring the user profile and responding to other services or applications requests. Depending on the type of the request, this module can use functionality offered by the Dynamic Modeler or to the Multi-Profile Aggregator in order to fulfill the request.

(b) The Static Modeler, which is responsible for the creation, removal and modification of user profiles at user's or application's request. A GUI will also be provided by this component, allowing the user to edit its personal data, enable modeling of certain preferences, or directly set preferences values.

(c) The Feedback Analyzer, which enable the update of the profile at system's initiative, based on explicit or implicit user feedback. This component will also provide functionality to request explicit user input, using the User Interface Service, whenever needed by the other components of UMPS.

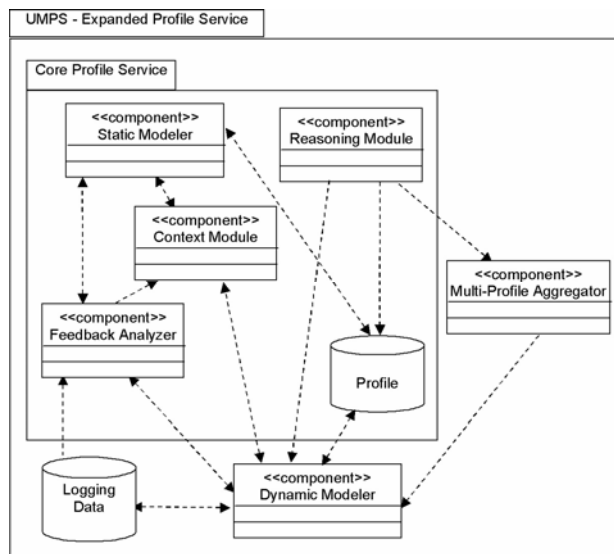
(d) The Context Module, which provides access to the context history data gathered by the Context Management Service. Both, asynchronous event-based subscriptions to context data and synchronous context queries are supported by this component.

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

The Multi-Profile Aggregator will provide an aggregated profile in case of multiple users found in the same context (i.e. the same room). Aggregated profiles can be obtained by merging profiles of several individuals whose preferences are known beforehand or by considering presence of other persons as social context.

The Dynamic Modeler will enable modification of the user profile data using the interaction/context history, the implicit/explicit feedback received from the user, as well as context changed events received through the Context Module from the Context Management Service. Depending on the data received, the profile can be updated by activating additional stereotypes, or by learning new parts of user profile (reasoning on context history data). Activation of new stereotypes may result in modification of the profile tree (enrichment with new preferences or removal of non-valid ones), change of preference values (merging of values from different

sources), and modification of system confidence on preference values.



**Figure 2 Architectural design of UMPS.**

The major research areas considered for modeling in Amigo are multimedia preferences and personalization/adaptivity of the speech user interface. Specifically, research on integration of personalization of speech user interfaces in the Amigo environment will consider: (a) use of stereotypes based on users' levels of expertise with speech interfaces, (b) use of correction grammars to correct speech understanding errors due to user specific pronunciation and use of language and (c) optimization of dialogue flow using system learning from interactions (reinforcement learning).

So far, CMS and UMPS provide the means to create highly adaptive, context-aware services or applications. Given the complexity of smart home environments, however, it is equally important to ensure that context information is filtered and transmitted to interesting parties in order to avoid having each possible entity deal with each potential piece of context information. This is handled by the Awareness and Notification Service.

## **AWARENESS AND NOTIFICATION SERVICE (ANS)**

The Awareness and Notification Service (ANS) enables developers to rapidly develop applications that allow people and other applications to stay aware of any significant change in context with minimal effort. Context that ANS is able to keep track can be of various nature such as, the activities, presence, and location of other people.

From the system perspective, ANS makes applications aware of context changes by notifying them. Applications register monitoring rules that specify what changes in context shall be notified to them.

From the user perspective, ANS provides context-aware notifications with appropriate rendering of intensity, based on the user's preferences and current context.

## **Rapid Development of Context-Aware Applications with ANS**

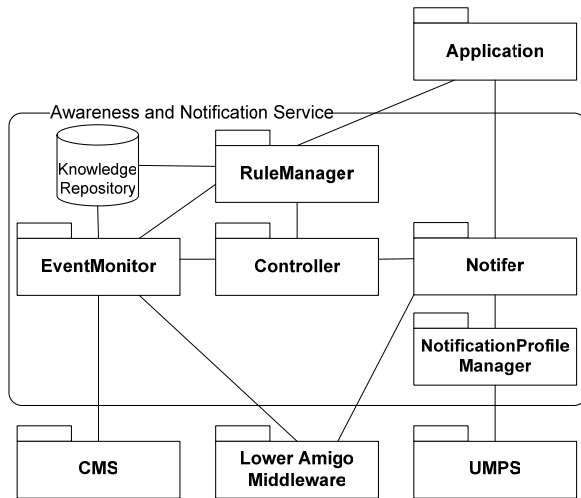
Building applications that allow people to stay aware of context changes with ANS is highly convenient. First the Amigo service discovery mechanism is used to discover ANS. ANS has one main interface that allows an application to install notification rules. A notification rule defines in which context a notification is sent to the application. For example, an application might define a notification rule that notifies Maria when her kids enter home with friends. Once this rule is set, ANS constantly evaluates this rule. As soon as the rule turns to true ANS notifies the application. The benefit of using ANS is that applications do not have to care about subscribing to and monitoring of context data. These tasks are handled by ANS. Furthermore complex contexts can be described conveniently by using the defined rule language. Another feature of ANS is, that it provides notifications based on the user's preferences and their current context. In order to be notified appropriately, users can create an individual notification profile. Based on this profile, ANS sends a notification with an appropriate rendering of intensity. The application receiving the ANS notification implements the notification of the user according to the intensity, for example the usage of an ambient light for an ambient notification.

## **Architectural Design**

Figure 3 shows the overall architectural design of ANS. The implementation of ANS is based on the principles of service oriented architecture. The service is implemented as a web service and is based on the standards SOAP, WSDL, XML, and UDDI. ANS has a clear defined interface that allows to manage and use of the service. All external services that ANS depends on are implemented as web services, too.

Internally ANS follows a component based approach with loose coupling between the components. The service consists of the 5 main components RuleManager, EventMonitor, Controller, Notifier, and NotificationProfileManager that are introduced in the following.

The RuleManager allows applications to manage notification rules. Applications set rules and so specify the contexts which shall be notified to them.



**Figure 3 Architectural design of ANS.**

The EventMonitor is responsible for finding and subscribing to the required context data. It interfaces neatly with the Context Management Service (see CMS) that manages the handling of the context sources.

The Controller contains a rule engine that is based on the rule engine Jess (8).

The Notifier is responsible to send notifications to applications and users with the appropriate intensity.

The NotificationProfileManager's task is to provide the notifications preferences of the users. It interfaces with the User Modeling and Profiling Service (see UMPS) that manages these profiles.

Internally ANS works as follows. First the RuleManager parses the rules. It extracts and relevant context data from the condition part of the rule and tells the EventMonitor to find and subscribe to the corresponding context sources. After parsing each rule, the RuleManager stores it in the rules database. Semantically and syntactically valid rules are processed by the Controller, which, in turn, receives event notifications from the EventMonitor. The Controller evaluates the rules and when a rule turns true, it checks the action part of the rule. In the action part it is specified which user or application is to be notified. The Controller then invokes the Notifier to send a notification.

Upon a request for notification, the Notifier asks the NotificationProfileManager for the respective notification profile. The Notifier then checks the profile to find out how the user should be notified. In order to get current context information about the user, the Notifier invokes the EventMonitor. The Notifier then evaluates the user notification profile against his current

context information and sends a notification with an appropriate intensity.

## The ANS Rule Language

Applications that want to be notified have to register monitoring rules. The developed ANS rule language follows the Event-Condition-Action (ECA) pattern. In this pattern, an Event models an occurrence of interest (e.g., a change in context); A Condition specifies a condition that must hold prior to the execution of the action; and an Action consists of requesting a notification service. The ECA pattern, or sometimes called Triggering pattern, has been widely explored in the field of context-awareness (Henricksen and Indulska (7), Ipina and Katsiri (6)).

When designing the ANS ECA rule language, high attention has been paid to the following qualities:

- Expressive power: the language permits the specification of complex event relations. It allows the use of relational operator predicates (e.g., < , > , =), and the use of logical connectives (e.g., AND, OR, NOT) on conditions to build compound conditions.
- Convenient use for application developers: It provides high-level constructs that facilitate event compositions.
- Extensibility: The language allows the addition of new predicates to accommodate events being defined on demand.

The example of the rule 'Maria would like to be notified when her kids enter home with friends' would be stated in this way:

```
Upon EnterTrue (Pablo.isAtHome) OR EnterTrue (Roberto.isAtHome)
When (Pablo.isAtHome AND Pablo.withFriends) OR (Roberto.isAtHome AND Roberto.withFriends)
Do Notify (Maria, "kids are home with friends")
```

## Context-Aware Notifications

User notification profiles enable ANS to notify users with the appropriate level of intensity. Before sending a notification to a user, the service checks the user's notification profile in order to determine the level of intensity. A user notification profile defines which level of intensity is appropriate in which user context. Available context parameters are for example the availability of the user that is to be notified, the location of the user that is to be notified, and the co-presence of other persons. The notification profiles are dynamic and allow users to take into account additional context data,

as soon as this data becomes available in the Amigo system.

When editing his user notification profile, a user is free to combine the above parameters in order to specify appropriate levels of intensity for certain situations. Each user can individually edit his/her profile by using UMPS, where the profiles are stored. In case the settings are conflicting, a conflict strategy implemented in ANS resolves the issue.

While ANS already deals with what kind of information to transmit to users or applications, the mechanisms of how the interaction with users should be performed is left to dedicated user interface services that integrate various interaction modalities common in smart home environments such as natural speech, gestures, or traditional graphical user interfaces.

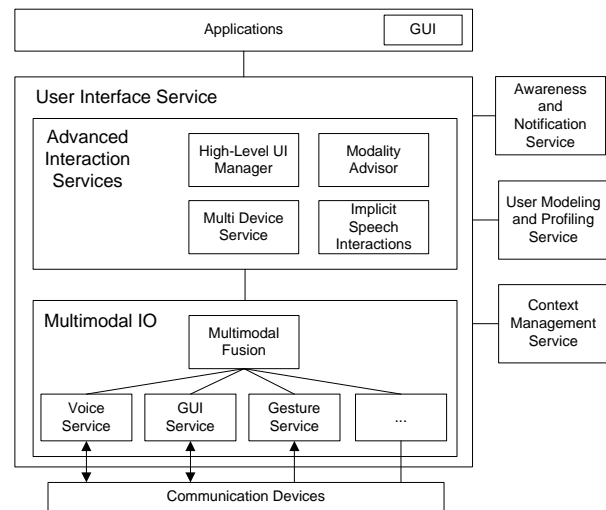
### USER INTERFACE SERVICES (UIS)

Our everyday interpersonal communication utilises several different senses such as hearing, speaking, sight and touch. The multimodal interaction, i.e. the combination of different interaction modes such as speech and gestures, starts right at our birth and it is natural and essential part of our everyday life. Recent development has enabled us to use multiple modalities also in human-computer interaction, making the communication more flexible, enjoyable and natural.

This chapter presents the User Interface Services that provides multimodal interaction framework for the Amigo Ambient Intelligent environment. The developed interaction framework combines natural and flexible end-user interfaces and communicates intelligently via several modalities or even with multilingual capabilities enabling the user to handle the overall home easily and by using the most appropriate and convenient interaction mode each time. The user interfaces designed for Amigo support dynamic adaptation to the users, context and devices. The User Interface Service consists of three different modalities, namely voice input and output, gestures and graphical user interface. Figure 4 presents the conceptual architecture of the User Interface Services.

User input to the system is captured using different communication devices. Graphical User Interface can utilise different mobile platforms such as PDA's, which can also be used for stylus based 2D gesture input. The concept of the GUI Service is to provide a platform which combines every given service to build a homogenous control interface of the environment. The actual GUI is automatically generated by the GUI Service exploring information provided by the services. Therefore each service has to explain its capabilities (functionalities) and the internal dependences. Voice IO is realised using microphone and speaker arrays spread

around the ambient intelligent environment. In addition to the typical speech recognition and synthesis voice can be also used for voice activity detection, speaker change detection and recognition, and acoustic source localisation and tracking. Finally the 3D gestures and infrared based pointing is captured using dedicated wireless control device equipped with acceleration sensors. Gestures can be used to select a control targets from the environment and to control those with predefined set of hand gestures. In addition to the typical explicit control these input channels can also be used for implicit interaction. For example audio input can be used to identify users and to recognise the topic of the discussion.



**Figure 4 Architectural Design of the User Interface Services.**

The inputs, i.e. communicative events, from different modalities are merged by Multimodal Fusion. The aim of the Multimodal Fusion is to combine the information provided by the user via its communication channels and to reduce the ambiguities and to finally resolve the actual meaning of the user. The merged output is delivered to the High-Level UI Manager that transmits the communication event to the corresponding applications. If the High-Level UI Manager doesn't know the target application for the event, i.e. the communicative event is ambiguous or it has no target application High-Level UI Manager initiates a dialog to resolve the problem. The High-Level UI Manager is also responsible of synchronisation of the dialogue. For example, if the user is programming the Personal Video Recorder functions by voice the selection the user has already made has to be also reflected in the Graphical User Interface dialogue, thus making it possible to switch between modalities inside the task.

Implicit Speech Interactions provides a generic framework to facilitate the work of the application developer who wants to develop applications that

continuously “listen to” the user and react accordingly. Ideally, everything the user is doing (movements, speech, gaze, etc.) that can be perceived by the system may be potentially interpreted as an implicit interaction. Such information perceived by the system is not directly exploited as an implicit interaction, but it is first included into the context; thus, the instantaneous context continuously evolves, and when it reaches some given state that can be interpreted as an implicit input. The corresponding Amigo service is activated and it may or may not react, depending on the user preferences or other contextual parameters. In Amigo, implicit speech inputs are considered, and the service that handles them will mainly exploit information about the topic of discussion of the user in specific application-dependent situations.

The Multi Device Service and Modality Advisor assist the Amigo applications in finding optimal compositions of modes and devices for the given context. By specifically addressing the uniqueness of different modes and devices, a dynamic, re-configurable, and context-aware composition of devices can be achieved. The 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 a 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 User Interface Service has also support for direct access to the modalities. In some cases applications might want to have private access to certain modality without using the Multimodal Fusion. For example games usually require fast uninterrupted access to the controller interface, thus making it feasible to keep the communication path as short and efficient as possible. In these cases a separate instance of the modality is used to serve the direct request, making it possible to use the same modality service for example in other room.

In this chapter an abstract architecture of the intelligent user interface services was presented. The main goal of user interface services is to provide a multimodal interaction framework that will be user-friendly and user-adaptable, will better support dynamic adaptation to the context and devices and will further support more natural implicit and explicit user interactions using multiple modalities, such as natural language and gestures.

The final service discussed in this paper addresses privacy and security. Given the all-importance of protecting private data acquired by the various sensors

deployed in a smart home, privacy & security has a special role among the discussed services. With regard to its pervasiveness, there is no single privacy service, but mechanisms that other services and applications can use to ensure privacy and security.

## PRIVACY & PERSONAL SECURITY (PPS)

The role of the Privacy and Personal Security Mechanisms is to ensure people’s 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, 2) and protected.

Privacy and personal security issues play a role in nearly all elements of the Amigo scenario. For example, the following situations affect the perception of privacy and security by users:

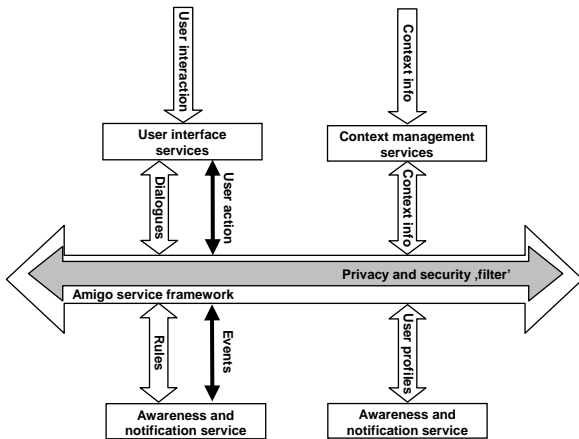
- Acquisition of surveillance data of users.
- Using implicit means for collecting information.
- Disclosing personal information to other people and services: preferences, habits, lifestyles, health information, location information, and availability.
- Storage risks: information might get lost, hidden,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

It is assumed that the Amigo system will 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. This special role of the Privacy and Personal Security Mechanisms requires a different design approach than the other Intelligent User Services. The diagram in Figure 5 shows this conceptual view.



**Figure 5 Amigo Service Framework.**

In Amigo, a distinction is made between *how* the appropriate levels of privacy and personal security within the Amigo Service framework can be ensured and accomplished and *what* the guidelines and rules are that can account for it.

With regard to the ‘how’, we envision in first instance a *PrivacyFilter* implemented as a rule-based system. With regard to what needs to be implemented to ensure the user’s privacy and personal security we take an empirical approach in which elements from the Amigo scenario (Amigo Deliverable, 2) will be selected. These elements provide the basis for cases that can be implemented or simulated and exposed to users. This user involvement is organized in a systematic and controllable fashion as to derive guidelines that can be represented in rules.

Since “perfect” privacy guarantees are in general hard to provide Chen and Kotz(3), Salber et al (13), the users of an Amigo system should be able to have the control over their contextual information and over who may gain access to it. The system architecture then needs to provide user-controllable tradeoffs between privacy guarantees and both functionality and efficiency. The difficulty then is to be specific about what context information should be visible to whom, and when.

The concept of privacy used for Amigo consists of a physical, informational and social dimension. For example, consider the situation when someone is at the door of an Amigo home. The physical dimension is determined by entrance rules that people maintain. The informational dimension consists of three factors, i.e., information sensitivity, information receiver, and

information usage. The social dimension is determined by interpersonal closeness and relationships.

In short, privacy is a dynamic and volatile concept, which is subject to significant variation depending on each individual and the current situation. To guarantee adequate privacy protection, individual privacy settings have to be dynamically adapted to the present context.

The conceptual data model is based on three concepts: *PrivacyGroup*, *PrivacyRules* and *PrivacyFilter*. *PrivacyGroup* is 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’. A service can use *PrivacyGroup* in specific ways. For example, from information acquired from the Context Management Service, *PrivacyGroup* can determine when the location of users is known and where this location is, from information acquired from the Content Storage and Distribution module in the Base Middleware, *PrivacyGroup* can determine ownership of content and from the User Modeling and Profiling Service, user profiles can be acquired.

*PrivacyRules* is 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*. 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. 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*.

The privacy mechanisms will be implemented orthogonal to the functionality of the other Amigo services. Since perceived privacy and security changes and varies dependent on context and persons, the Amigo project adopted a multifold approach based on generating empirical evidence obtained from users in actual situations, deriving guidelines and rules for privacy, and providing users with means to control their personal privacy conditions.

## CONCLUSIONS

In this paper, we presented the initial Amigo intelligent user services that allow for creating context-aware applications for smart home environments. The discussed composition of services facilitates both

people-oriented and system-oriented approaches. While Amigo aware applications that integrate context management and user modeling services are capable of operating rather autonomously and adaptive to user and context information, the sophisticated user interface services as well as the awareness and notification service are provided explicitly to keep the user in the loop. The degree to which an application can be more people-oriented or more system-oriented is largely determined by the respective Amigo services it integrates.

It is a matter of experimentation whether people- or system-oriented approaches are more beneficial in the real world situations of smart home environments. We believe that a sensible mixture of both paradigms provides the most efficient and acceptable user experiences. The exact degree to which both approaches should be realized will be investigated within the Amigo project by using the intelligent user service infrastructure as a testbed for respective user studies. Several living laboratories that will host the Amigo service infrastructure are already set up at partner sites. We now work towards the first distributable suite of the intelligent user services, so that the aforementioned user studies can be conducted and also feedback can be gathered from users outside the Amigo project.

## ACKNOWLEDGEMENTS

The authors would like to thank all their colleagues and co-workers from the eleven research groups involved in the Amigo IUS work package who actively design, implement, and test the services discussed above. Special thanks go to Tom Broens, Christophe Cerisara, Patricia Dockhorn Costa, Henk Eerting, Sanna Kallio, Basilis Kladis, Remco Poortinga, Thorsten Prante, Matthieu Quignard, Christian Ressel, Maja Stikic, Elena Vildjiounaite and Peter Vink for their involvement in the creation of this paper.

The Amigo project is funded by the European Commission as an integrated project (IP) in the Sixth Framework Programme under the contract number IST 004182.

## REFERENCES

1. Amigo Project – <http://www.hitech-projects.com/euprojects/amigo>
2. Amigo Deliverable D1.2: Report on User Requirements, Janse M. D. (ed.), IST-004182 Amigo, April 2005
3. Chen, G., Kotz, D., A Survey of Context-Aware Mobile Computing Research. Dartmouth Computer Science Technical Report, 2000

4. Dey, A. K., Salber, D., Abowd, G., A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16, pp. 97-166, 2001.
5. Fink, J., Kobsa, A., User Modeling and User-Adapted Interaction, 10, pp. 209-249, 2000.
6. Ipiná, D., Katsiri, E., An ECA Rule-Matching Service for Simpler Development of Reactive Applications. Published as a supplement to the Proc. of Middleware 2001 at IEEE Distributed Systems Online, Vol. 2, No. 7, November 2001.
7. Henricksen, K., Indulska, J., A software engineering framework for context-aware pervasive computing. Proc. of the 2nd IEEE Conference on Pervasive Computing and Communications (Percom2004), Orlando USA, 2004.
8. Jess – the Rule Engine for the Java Platform. Available at <http://herzberg.ca.sandia.gov/jess/>
9. Khedr, M., Karmouch, A., ACAN-Ah hoc Context Aware Network. IEEE CCECE/02, Winnipeg, Canada, 2002.
10. Nebel, I. T., Smith, B., Paschke, R., Tagungsband der GI-Workshopwoche Lernen – Lehren – Wissen – Adaptivität, Universität Karlsruhe, 327-330, 2003.
11. Papaioannou, I., Tsesmetzis, D., Roussaki, I., Anagnostou, M., A QoS Ontology Language for Web-Services. Accepted for AINA 2006.
12. Rich, E., Readings in Intelligent User Interfaces. Springer-Verlag, Berlin, Germany, 1998.
13. Salber, D., Dey, A. K., Abowd, G., The Context Toolkit: Aiding the Development of Context-Enabled Applications Proceedings of the 1999 Conference on Human Factors in Computing Systems (CHI '99), May 15-20, 1999, pp. 434-441.
14. Spreitzer, M., Theimer, M., Providing location information in a ubiquitous computing environment. Proc of the 14th ACM Symposium on Operating Systems Principles, (270-283), Asheville, NC, December 1993.
15. Streitz, N. A., Röcker, C., Prante, Th., van Alphen, D., Stenzel, R., Magerkurth, C., Designing Smart Artifacts for Smart Environments. In: IEEE Computer, March, 2005. pp. 41-49.
16. Vallée, M., Ramparany, F., Vercouter, L., Flexible composition of smart device services. In: The 2005 International Conference on Pervasive Systems and Computing (PSC-05), Las Vegas, Nevada, USA., June 27-30, 2005.