Project Identification	
Project number	AAL-2010-3-070
Duration	1st July 2011 – 30th June 2014
Coordinator	Christopher Mayer
Coordinator Organisation	AIT Austrian Institute of Technology GmbH, Austria
Website	www.aaluis.eu

Ambient Assisted Living user interfaces

Document Identification	
Deliverable ID:	D2.1 Report on Middleware Analysis
Release number/date	V1.0 / 30.12.2011
Checked and released by	Martin Morandell/AIT

Key Information from "Description of Work"	
Deliverable Description	This document describes the criteria and findings of the middleware analysis, as well as findings of research on personalization and user (interaction) profiling.
Dissemination Level	PU=Public
Deliverable Type	R = Report
Original due date	Project Month 4 / 31. October 2011

Authorship& Reviewer Information	
Editor	Kai Hackbarth / ProSyst
Partners contributing	AIT, PHIL, VERK
Reviewed by	Miroslav Bojic / PHIL

Release History

Release Number	*Date*	*Author(s)*	*Release description /changes made*
V01	15.02.2011	MMo/AIT	First version of Del Template
V02	15.05.2011	MMo/AIT	Title Page, Contract Number, some Structure elements
V03	06.10.2011	MG/AIT	Content
V04	19.10.2011	HH/VERK	Content
V05	19.10.2011	PHIL	Content
V06	19.12.2011	PHIL	Internal Review
V1.0	30.12.2011	KH/ProSyst	Contribution by all partners and final editing

AALuis Consortium

AALuis (AAL-2010-3-070) is a project within the AAL Joint Programme Call 3. The consortium members are:

Partner 1	*AIT AUSTRIAN INSTITUTE OF TECHNOLOGY GmbH (AIT, Project Coordinator, AT)*
Contact person:	Christopher Mayer
Email:	christopher.mayer@ait.ac.at
Partner 2:	*weTouch e.U. (weT, AT)*
Contact person:	Christian Schüler
Email:	Christian.schueler@wetouch.at
Partner 3:	*Centre for Usability Research & Engineering (CURE, AT)*
Contact person:	Jan Bobeth
Email:	bobeth@cure.at
Partner 4	*zoobe message entertainment GmbH (Zoobe, DE)*
Contact person:	Sascha Fagel
Email:	Fagel@zoobe.com
Partner 5	*Verklizan BV (Verk, NL)*
Contact person:	Matti Groot
Email:	mgroot@verklizan.com
Partner 6	*ProSyst Software GmbH (PRO, DE)*
Contact person:	Kai Hackbarth
Email:	k.hackbarth@prosyst.com
Partner 7	*50plus GmbH (50plus, AT)*
Contact person:	Tanja Bosch
Email:	tanja.bosch@seniorenbund.com
Partner 8	*Hilfswerk Österreich (HWOe, AT)*
Contact person:	Walter Marschitz
Email:	walter.marschitz@hilfswerk.at
Partner 9	*Philips Consumer Lifestyle B.V. (PHIL, NL)*
Contact person:	Kees Tuinenbreijer
Email:	kees.tuinenbreijer@philips.com

Table of Contents

Table of Figures

List of Tables

Abbreviations

Abbrev.	*Description*
AAL	Ambient Assisted Living
AAL JP	Ambient Assisted Living Joint Programme
AALuis	Acronym of this Project – Ambient Assisted Living user Interfaces

Executive Summary

AALuis intends to develop a User Interface Layer that facilitates the connection of any kind of service with different types of user interfaces, such as smartphones, tablets, TVs, PCs or voice. In the past years there has been a lot of funding in the development of middleware technologies in the AAL domain, which is why AALuis will not develop its own middleware. The aim of the AALuis User Interface Layer is to be as much independent of the middleware technologies as of the user interfaces itself.

Work package 2 conducted a research on what we think are the most promising AAL middleware technologies and how we can make use of them in the context of AALuis. Besides a general description and an overview of the layer model, we also defined various criteria that formed a basis for analysis of AAL middlewares. Examples of these criteria are the status of the project/technology, openness in terms of how easy it can be used by others, functionality richness, how communication with devices and services work. In addition to AAL middleware technologies that were developed in research projects we also analysed the middleware technologies provided by our project partners Philips and Verklizan.

Work package 2 also made a state-of-the-art analysis on personalization and user profiling as we want to design and develop a user interface layer that takes both aspects into account.

1 About this Document

1.1 Role of the document

This document provides an overview of Ambient Assisted Living middlewares' developed in various research projects as well as from AALuis partners Philips and Verklizan. The technical analysis compares AAL middlewares by looking into aspects such as general architecture, interoperability with external systems / services as well as whether and how the AALuis User Interface Layer can be built on top.

1.2 Relationship to other AALuis deliverables

The deliverable is related to the following AALuis deliverables:

Deliv:	*Relation*
D2.2	Handbook AALuis Layer Specification: this document specifies the AALuis User Interface Layer and how it can be integrated with already existing AAL middlewares' described in D2.1.
D3.1	Report on User Interface Analysis: this document provides an overview of innovative user interfaces for ambient assisted living environment. Complementary analysis in WP3.
D4.1	Report on Service Application Analysis: this document provides an overview of innovative ambient assisted living service applications. Complementary analysis in WP4.

2 Criteria for the Middleware Analysis

Middleware analysis criteria	
Status	• What is the current status of the project/technology? • What is commercial impact of the project/technology?
Openness	• Availability of the results incl. terms and condition as well as licenses • Is there a e.g. developer community and how are they supported? • How well is the technical documentation?
Richness	• Overview of most important functionalities • What are interesting / important components in the context of AALuis?
Communication	• How is service and/or UI discovery realized? • How can AALuis be integrated?

Table 1: Template for Middleware Analysis:

In addition we created a table for each of the middlewares to describe their layered architecture. The cell “AALuis use” is intentionally empty for now. These cells will be filled when AALuis architecture has been designed.

3 Analysis of available AAL Middlewares

3.1 AMIGO

AMIGO was a research project funded by the 6th Framework Programme from the European Commission. AMIGO developed a middleware that allowed dynamic integration of heterogeneous systems to achieve interoperability between services and devices. The high-level architecture can be found below:

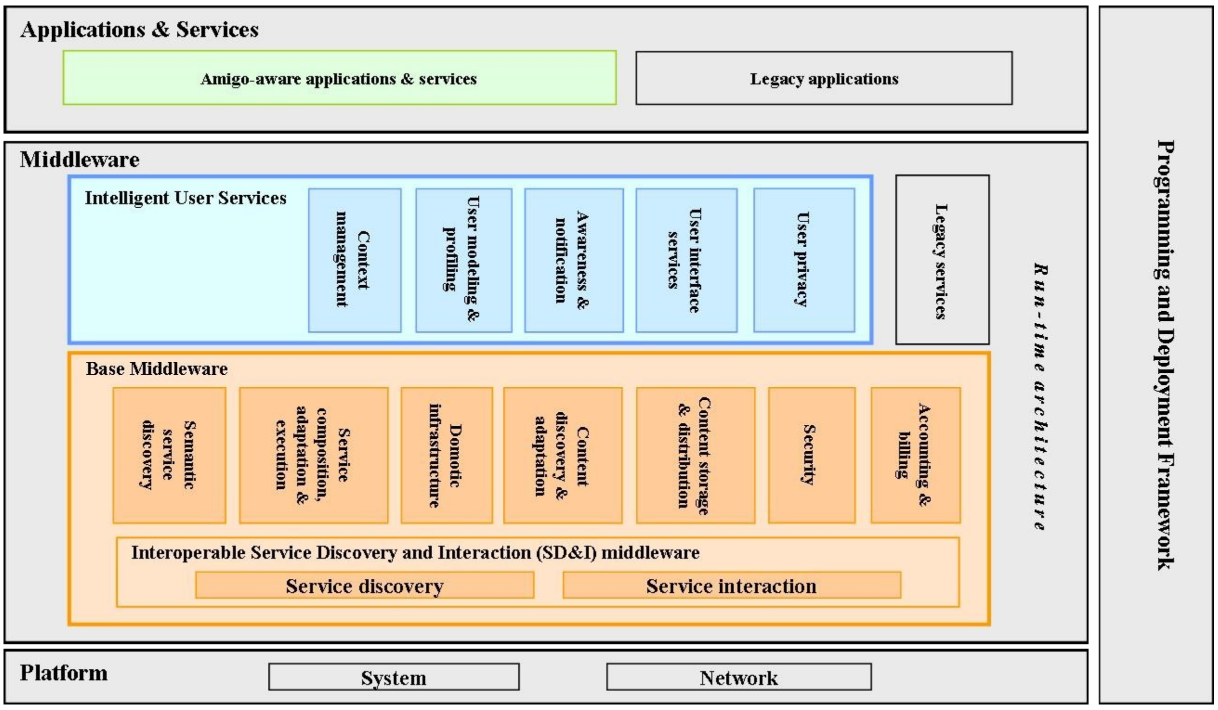

Applications & Services
Amigo-aware applications & services
Legacy applications
Middleware
Intelligent User Services
Context management
User modeling & profiling
Awareness & notification
User interface services
User privacy
Legacy services
Base Middleware
Semantic service discovery
Service composition, adaptation & execution
Domotic infrastructure
Content discovery & adaptation
Content storage & distribution
Security
Accounting & billing
Interoperable Service Discovery and Interaction (SD&I) middleware
Service discovery
Service interaction
Run-time architecture
Platform
System
Network
Programming and Deployment Framework

Figure 1: Building Blocks of the AMIGO architecture

3.1.1 AMIGO Layer model

Layer	*Description*	*AALuis use*
Applications & Services	Amigo-aware applications and services exploit the full functionalities of the underlying base middleware. Legacy services do not have any knowledge of the AMIGO middleware layer, but are implemented on some legacy service service oriented middleware.	
Base Middleware	Provides the semantics to communicate and discover available services and devices in the network. Supports technologies like UPNP, WS, or SLP. Security mechanisms for authentication, authorisation, and encryption are provided as well.	
Intelligent User Services	A broker provides context information and combines information from multiple source. Based on this, pattern-based predictions will be done. The Information is tailored to user profiles and adapts taking into the account the user's situation as well as the context.	
Programming and Deployment Framework	Facilitates the development of AMIGO-aware services in either .NET or Java. AMIGO abstracts several protocols for communication and discovery, so that service can be integrated independent from the underlying hard- and software technologies	

Table 2: AMIGO Layer model

Status:

AMIGO was an Integrated Project funded by the 6th framework of the European Commission. The project consortium consisted of the following partners:

- VTT
- Orange
- Inria
- University of Paderborn
- Fraunhofer SIT
- Fraunhofer IMS
- Microsoft
- SingularLogic
- ICCS
- Giugiaro
- Ikerlan
- Telefonica
- Fagor
- Philips (Project Coordinator)
- Telematica Institute

The project had a duration of 3½ years and finished in 2007. There is no evidence whether any of the results have been reused in other research projects or product developments.

Openness:

Some of results have been released as open source (incl. software developer and users guides) and can be downloaded from the Inria website. Most of the components are under LGPL open source license. Some components have a Microsoft and Philips specific license that allows their use for non-commercial activities.

Richness / Functionality:

AMIGO did not develop any AAL specific components or services. The scenarios were focussing in the areas of home information and entertainment, home care and safety as well as extended home environment. Functionalities implemented by the project including context management service, awareness and notification, privacy and security, user modelling and profiling, content distribution, accounting and billing, as well as user interface services.

Communication:

Service discovery is realized by the standardized mechanisms provided by the OSGi middleware and .NET platform. Besides this the project implemented an interoperable service discovery and interaction middleware that converts incoming/outgoing messages from protocol to another. Supported Technologies are UPnP, SLP, Web Services, SOAP and RMI.

3.2 MonAMI

MonAMI was a project funded by the by the EU 6th framework programme. Its main objective was to demonstrate that accessible, useful services for elderly persons and persons with disabilities living at home can be delivered in mainstream systems and platforms. [24]

The architecture of the main building blocks of the MonAMI system is displayed below:

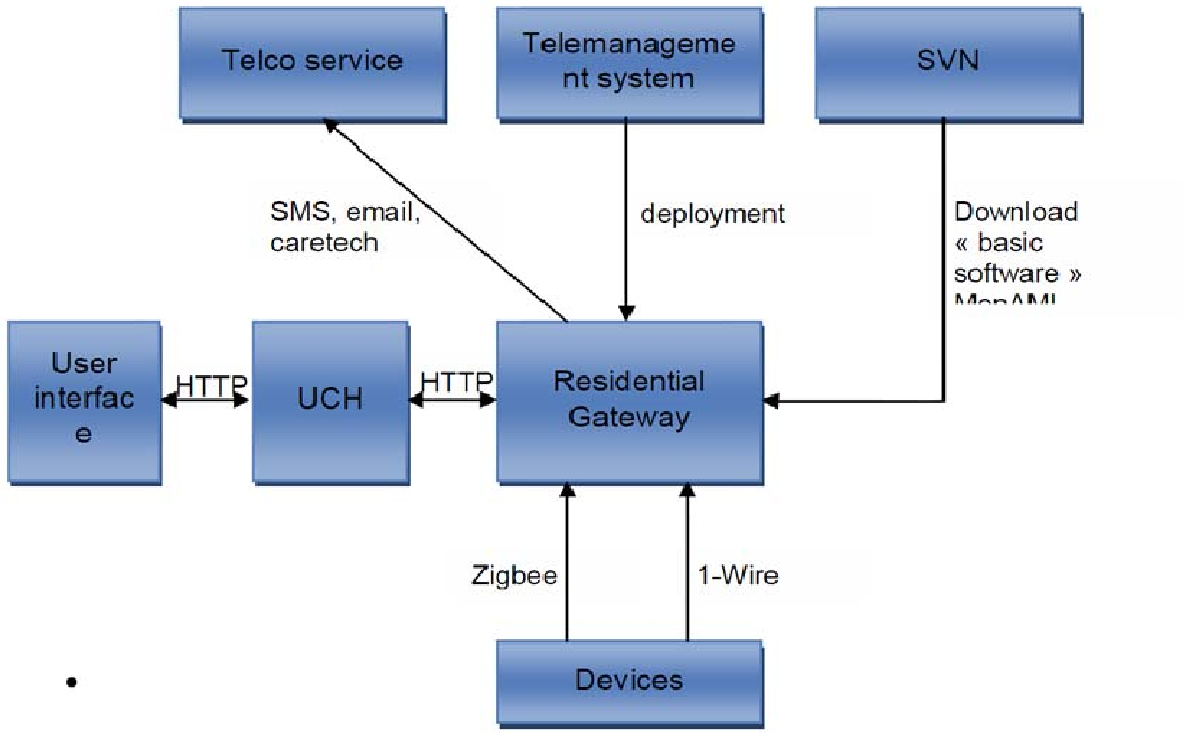


Figure 2: Building Blocks of the MonAMI architecture [24]

The building blocks are:

- A (off site) web service platform hosting telecommunication services, including SMS, E-Mail and other care technology.
- Graphical User Interfaces, which are presented to the user over a Universal Control Hub (UCH) server.
- Sensor devices for context information gathering and,
- Actuator devices for home environment manipulation.
- A SVN version control, used for integration and deployment of the system.
- The Residential Gateway (RG) hosting the MonAMI services.

Residential Gateway (RG)

The RG is considered the centre of the MonAMI system. Technically it is a Java Virtual Machine, hosting an OSGi framework. Services are implemented as OSGi bundles, running in the framework.

The RG architecture can be divided into four main parts:

- Framework: The framework consists of the bundles of the OSGi framework itself, providing base-line utilities, and the OSGI4AMI bundle. This bundle contains a common set of Java interfaces, which describe an ontology that is used as a conceptual basis for internal service communication.
- COM: Hosting bundles to communicate with external entities. These include sensors and actuators accessed over Zigbee or 1-Wire connections, telecommunication and telemanagement, and the UCH accessed over IP connections.
- Technical Services: Low level services of the MonAMI system, for example: alarm sender, configuration, and utility services used by higher level services.
- Functional Services: Higher level services of the MonAMI system. These include AMiVUE for detecting context information like device status and movement; AMiSURE for safety and security functions, abnormal situation detection, notifications and alarms; AMiCASA for home automation functionality like light switching.

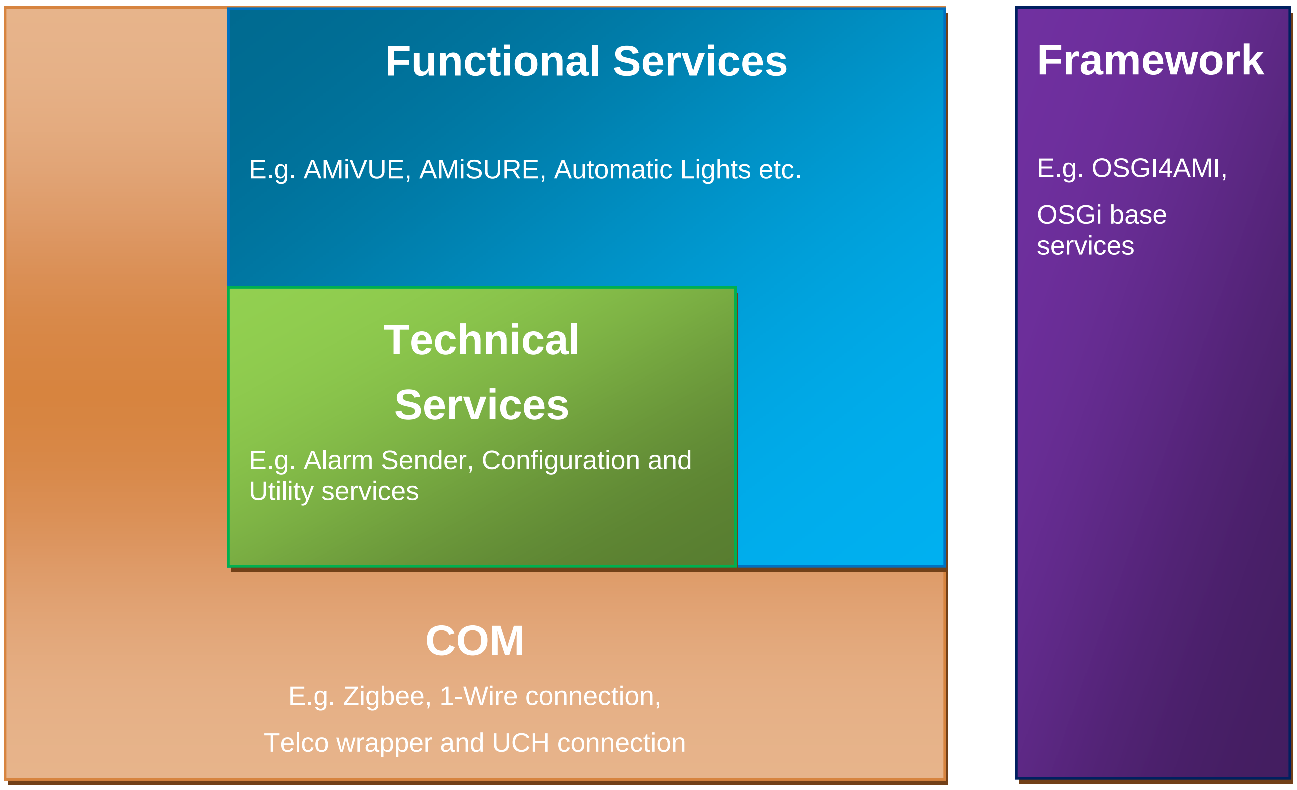

Functional Services
E.g. AMiVUE, AMiSURE, Automatic Lights etc.
Technical Services
E.g. Alarm Sender, Configuration and Utility services
COM
E.g. Zigbee, 1-Wire connection,
Telco wrapper and UCH connection
Framework
E.g. OSGI4AMI,
OSGi base services

Figure 3: The MonAMI platform

User Interaction

MonAMI recognises three different user groups: beneficiaries with various disabilities, carers, and developers. Each of them has a distinct user interface. The catering of these is provided via the Universal Control Hub (UCH) middleware, acting as a gateway between the user interfaces (UI) and the Residential Gateway (RG):

Main components of the user interface architecture:

- Beneficiary UI: A web based client called “QiWebclient” UI.
- Carer UI: Carer part of the UI, a web client accessed by an iPhone
- Developer UI: Auto-generated user interfaces, directly from the UCH.

- UCH: Universal Control Hub, a reference implementation of the Universal Remote Console (URC) standard, also described in this document.
- RG: The residential gateway hosting the core MonAMI services, described above

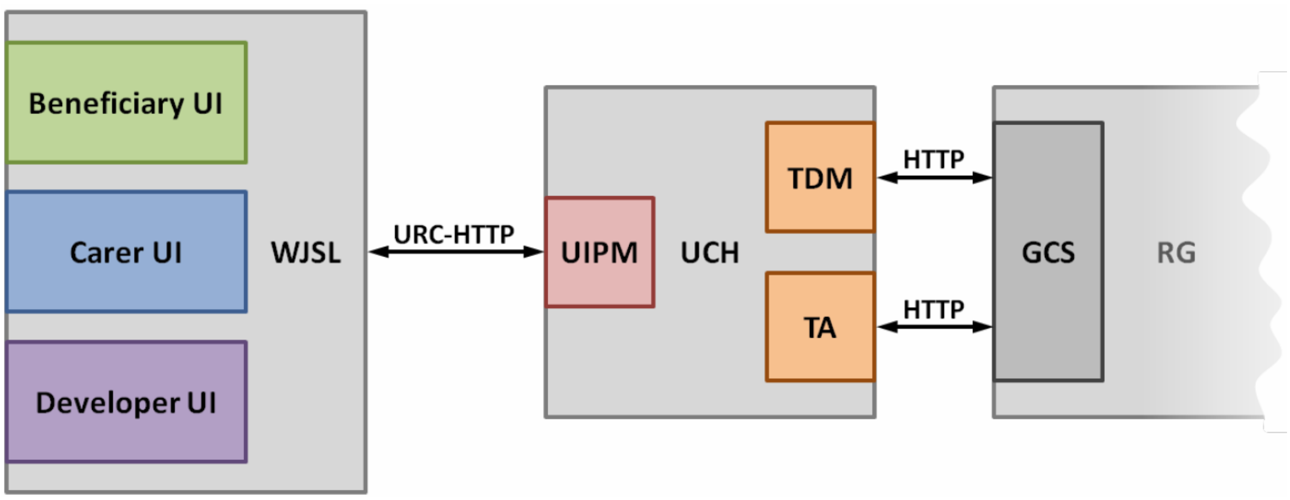


Figure 4: User Interfaces in the MonAMI project [25]

3.2.1 MonAMI Layer model

Layer model of the residential gateway:

Layer	*Description*	*AALuis use*
COM Layer	Communication Layer. Sensor and Actuator access over Zigbee A, Zigbee B, 1-Wire technology. Holds also the component for communication with the UCH server, and accessing telecommunication services.	
Technical Service Layer	Containing low level services of the MonAMI system.	
Functional Service Layer	Containing higher level services of the MonAMI system	
Framework	OSGi platform and services utilised in other Layers. Contains also the OSGi4AMI bundle, defining the MonAMI java interfaces for inter-component communication.	

Table 3: MonAMI Layer model

Status:

The project MonAMI is a project funded by the by the EU 6th framework programme, with a consortium constituted of the following partners:

- Swedish Institute of Assistive Technology (Competence Centre)
- Electricité de France (Electricity Utility)
- Europ Assistance France (Service Provider)
- France Telecom (Telecom)
- HMC International (Assistive Devices)
- London School of Economics (Research)
- OpenHub (Services)
- Royal Institute of Technology (Research)
- Siemens IT Solutions and Services (Consumer Devices)
- Technical University of Košice (Research, Training)
- Telefónica I+D (Telecom)
- Trialog (Software, SME)
- University of Passau (Research)
- University of Zaragoza (UZAZ) Research

The project started on September 1st 2006, and closed on May 31st 2011. After the project closure, the results remained hosted by, the coordinator, but no further endeavours are planned or to be expected. [26]

No commercial exploitation of the project was announced on available sources.

Openness:

All source code and documentation was released under an open source license. The release contains thorough documentation on the setup process, as well as source code documentation. This created no traction in the open source community, with no further active development visible.

Richness / Functionality:

The project created some, but not all planned, AAL services. In the domain of user interaction, a communication adapter to the UCH was created. This can be a valuable input and starting point for the AALuis project, if the usage of the URC technology is planned.

The UCH communication component could be a valuable input for AALuis. Other components and services appear to be project centric, and not universally applicable

Communication:

The UCH communication component connects the MonAMI platform the UCH world. In the project this was done to cover all user interaction needs. User Interfaces therefore are connected to the system over the URC-HTTP protocol to the rest of the MonAMI system.

In the OSGi based runtime, service-discovery and service-connection are done via mechanisms the OSGi framework provides. The only component of the MonAMI that connects via IP to external services is responsible to connect to telecommunication services. For non disclosed reasons this component was not part of the released source code, and cannot be analysed for further use.

Inter-component message exchange is realised through OSGi facilities. The basis of this, internal communication builds an ontology manifested in Java interfaces. Context information is held and processed in the service bundles.

User interaction of the service bundles is achieved through abstracted UIs that are provided through the UCH to the user interfaces.

3.3 Universal Remote Console, Universal Control Hub

The Universal Remote Console (URC) is an international standard (ISO/IEC 24752) defining a way to control arbitrary electronic devices or services (hardware or software) with interoperable, pluggable user interfaces that adapt or are adaptable to fit the user's special needs.

The standard distinguishes so-called "controllers" hosting the user interfaces, which control services and devices called "targets" in its context. In the URC system controllers and targets can interact without prior knowledge of each other. As a common knowledge the standard specifies XML formats to describe the entities of the URC ecosystem, which utilise the ISO 15836:2003 standardised Dublin Core metadata element set.

Each target exposes its functionality and state through one or more "User Interface Sockets" (UIS), specified in a special XML format and defined in the sub standard ISO/IEC 24752-2 [22]. These User Interface Socket descriptions formalise the capabilities of a target in a machine-interpretable abstract way, independent of any specific implementation platform. Additional information (e.g. Interaction mechanisms, resources for labelling, static components) is necessary to form a concrete user interface in the controller. This information is made available through additional XML, so called "User Interface Implementation Description" (UIID), as defined in the standard. An example for an UUID would be the "Presentation Template" (PT) [23]. It maps elements of a user interface socket to interaction modality independent intents. As a central point, for all of the information a target is advertising, a "Target Description" (TD) document is to be published by every target.

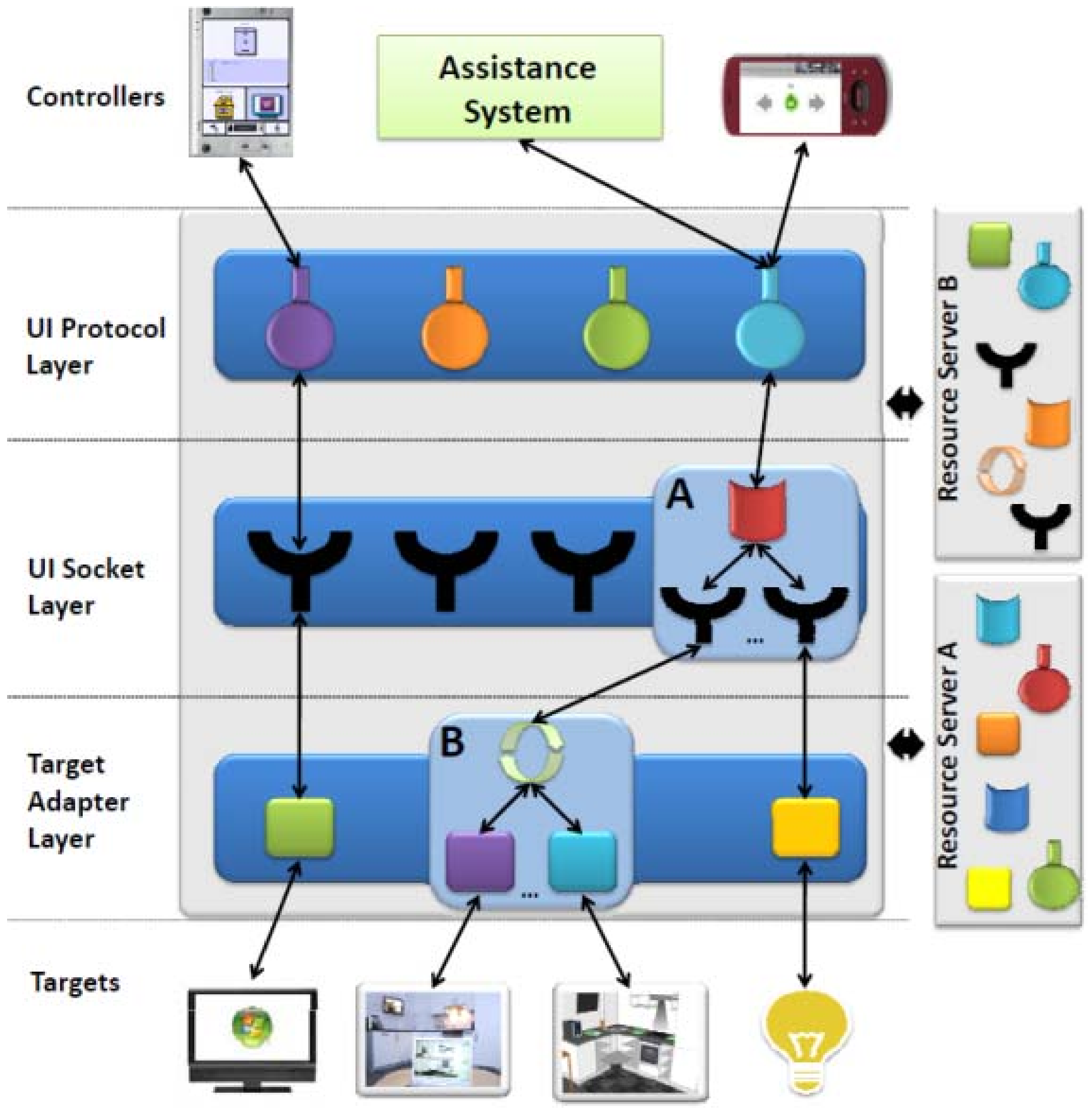


Figure 5: Layers of the URC architecture

The Universal Control Hub (UCH) is a profiling of the URC standard, to overcome the shortcomings of the URC in the area of mass adaption[1]. It realizes the URC standard as a middleware server component, providing connection points to existing, non URC compatible targets and controllers. UCH leverages prevalent network enabled devices and technologies, by using custom made proprietary components for discovery and control of targets.

[1] The URC standard needs in practice targets and devices that fully implement the URC information and communication model. As of now there is no significant adoption of the standard by manufacturers.

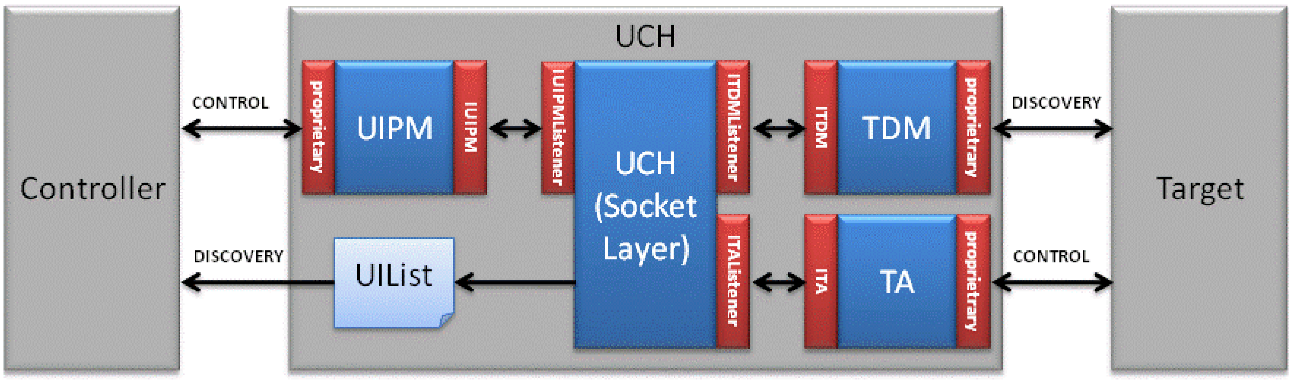


Figure 6: Logical view of the UCH architecture [19]

As proposed in [20], targets are exposed to the UCH framework through the UPnP mechanism. These are UPnP enabled devices of diverse provenance, communicating with UPnP enabled components of the UCH. Concrete user interfaces are catered by the UCH to clients. They use socket descriptions, which hold abstract interaction information, together with a Pluggable UI. The Pluggable UI contains concrete layout and rendering information, and can be served from a remote resource server.

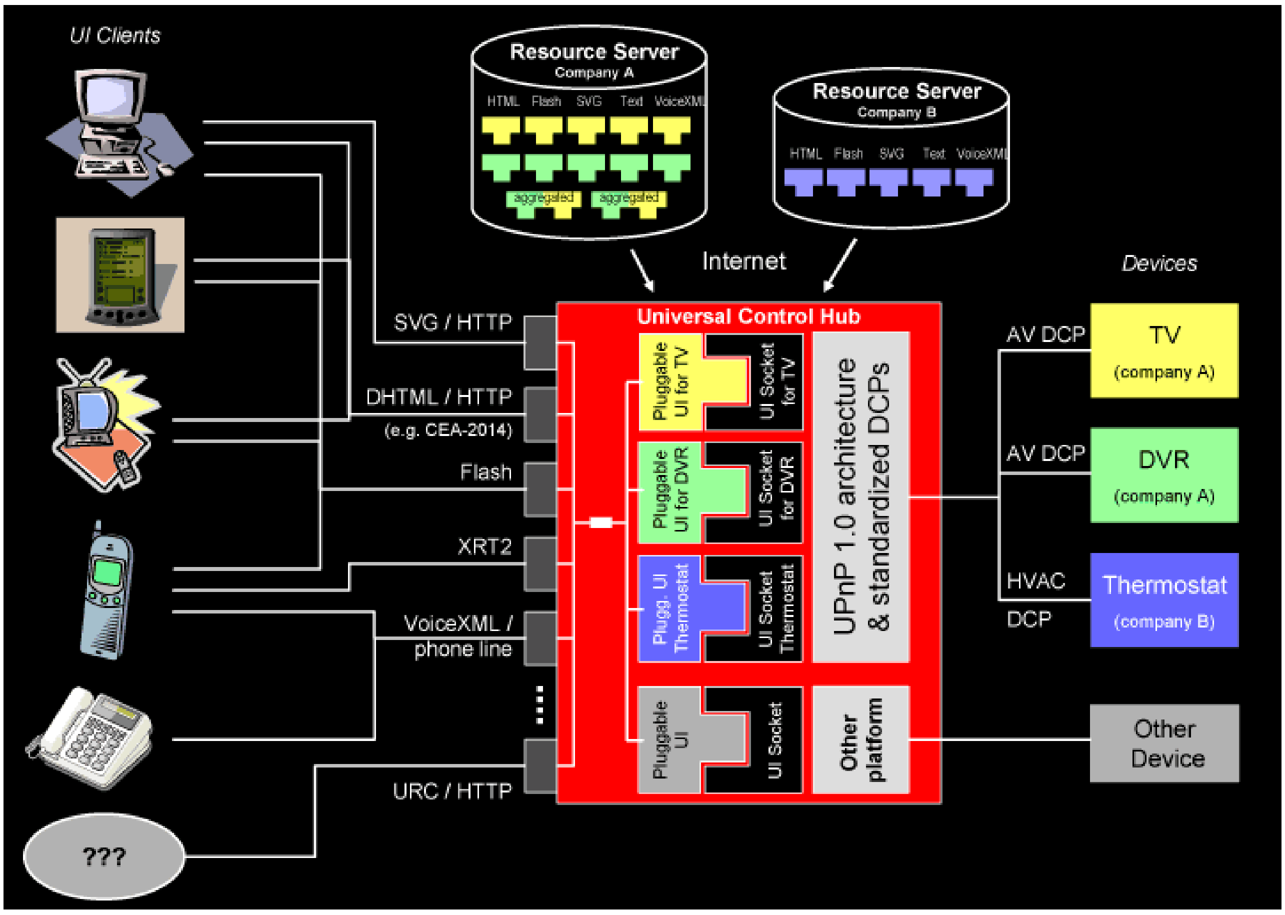


Figure 7: UCH architecture [20]

3.3.1 URC Layer model

Layer	*Description*	*AALuis use*
Target Layer	Services and devices to be controlled. Responsible to advertise its socket descriptions and resource sheets to the URC system through various XML files.	
Target Adapter Layer	UCH discovers a target through a target discovery manager, and controls it and receives events from it through a target adapter.	
UI Socket Layer	Holds abstract (User Socket description) and concrete (Pluggable UI) information about the user interface. To be rendered by controller user interfaces.	
UI Protocol Layer	Communication layer over which a controller is connected to the URC. E.g. URC/HTTP. UCH also describes other access methods like SVG/HTTP or DHTML/HTTP	
Controller Layer	User interface clients that present the concrete interfaces of known targets to the user, by any modalities it is capable of.	

Table 4: URC Layer model

Status:

The URC specification was approved as an international standard in 2008. Implementations of the standard are currently underway by the URC consortium.

There are currently three concrete implementations of the UCH, two of them open source: UCHj (Java implementation), UCHe (C/C++ implementation for embedded systems), and one proprietary.

While there is an initiative called openURC, which tries to propagate the URC standard, the open source projects UCHj and UCHe are not under active development since May 21, 2010.

Openness:

The URC standard documents the architecture and technical background thoroughly. UCHj and UCHe installation and configuration instructions can be found on the project home page.

Richness / Functionality:

No means to integrate user profiles or preferences into the user interface adaption or election.

Communication:

The service (target) and user interface (controller) discovery process is defined through the URC standard. User Interfaces that fully follow the URC standard are connectable by definition.

The UCH can be connected to any UPnP enabled device. Also network enabled devices could be connected via proprietary connectors.

Services can be understood as "targets", if they expose their user interaction through the standards XML descriptions they can be used. As in the MonAMI project, services that do not follow the standard per se need to be connected through "translation layers".

The UCH communicates with controllers through "User Interface Protocol Modules". Controllers aware of the URC standard, can directly access the user interface sockets, running in the UCH, through URC/HTTP – an HTTP-based messaging protocol. Other access technologies and protocols are possible (e.g. access to sockets through SVG/HTTP).

3.4 UniversAAL

UniversAAL is an ICT project funded under the 7th Framework Programme. It aims to produce an open platform that makes it technically feasible and economically viable to conceive, design and deploy innovative new AAL services for developers. End-Users will be addressed with a simple to use solution to download and setup these AAL services.

The developed platform is a mixture of new development and consolidation of existing platforms of the projects: AMIGO, GENESYS, MPOWER, OASIS, PERSONA, and SOPRANO.

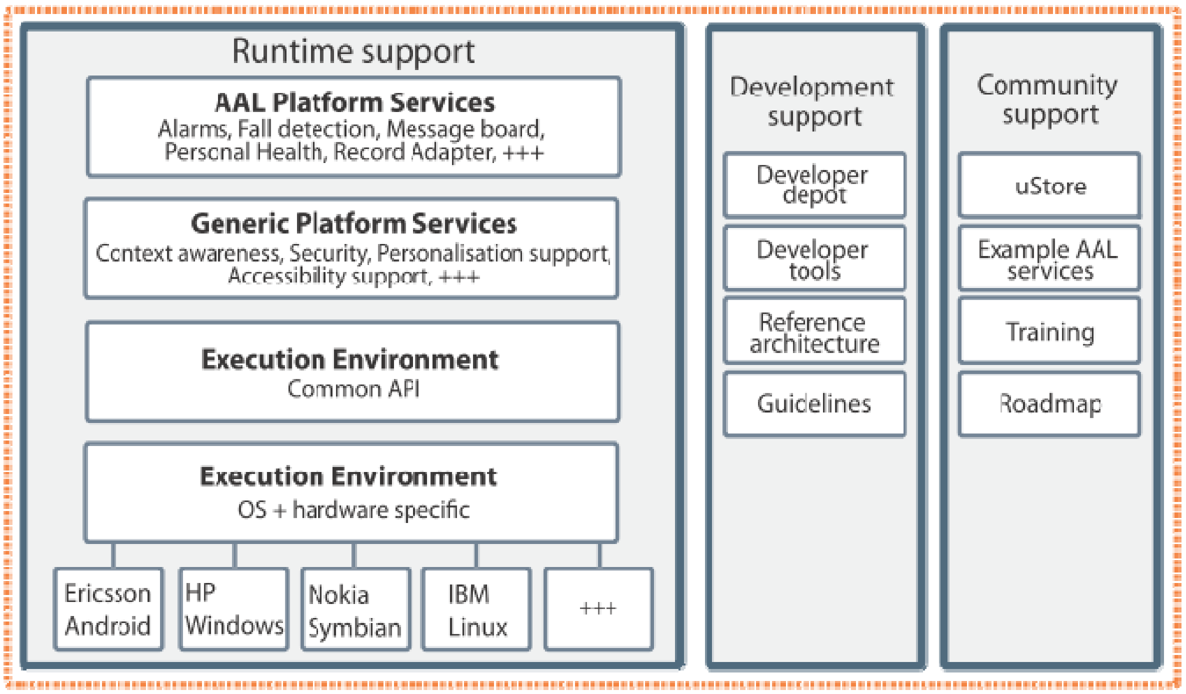


Figure 8: Platform from [27]

Based on the technical requirements, as described in [1], a reference architecture was created, and described in [2].

3.4.1 UniversAAL Layer model

The universAAL reference architecture structures the platform in a loosely layered way. There are distinct layers, but not in a strict way as, for example, the OSI layer model does. Instead each layer holds functional groupings of components, which are distributed over several networked nodes in the universAAL ecosystem. While every (middleware) service is transparently accessible on every node, not every middleware service or component is present on every node. Only the so called "Middleware" layer has an equivalent on every node enabling inter node communication.

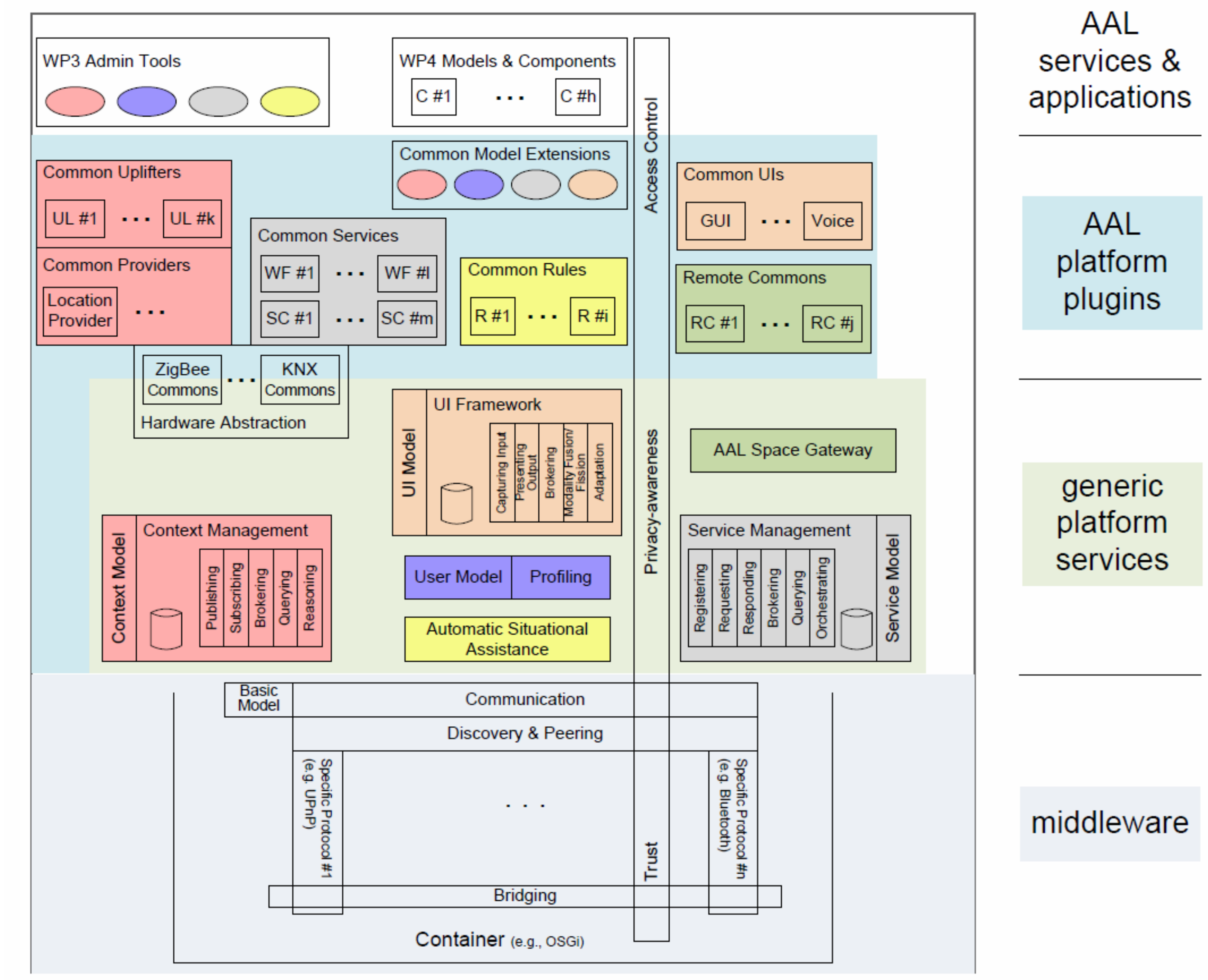


Figure 9: The UniversAAL architecture as described in [2]

Layer	*Description*	*AALuis use*
Middleware Layer	Has *footprints* across all nodes in a given running universAAL ecosystem. Understood as an extension to the native runtime environment of the node – also named the "Container". Discovery and peering. To hide the distribution of nodes they first have to be discovered. This is also covered by the middleware layer, in a protocol-flexible way. All communication has to be run over the middleware layer, to realise a great freedom in distribution of	

	services. The “Basic Information Model”, necessary to create correct and meaningful communication.	
Generic Platform Services Layer	Hardware Abstraction, binds network enabled nodes that are not AAL system “aware” by them self. For example proprietary sensory systems of the home environment. Context Management, provide adequate communication between providers and consumers of context data. Service Management, to facilitate service-based interoperability. I.e. registration of services, brokering of inter service requests. Profiling, for managing shared data representing user preferences and characteristics. User Interaction Framework addresses challenges related to explicit interaction between AAL spaces and its human users. Hides the complexity of I/O infrastructure from the upper application layers. Automatic Situational Assistance, automatic reaction of the AAL system to certain situations. AAL Space Gateway, contains basic means to open the AAL space to the outside world. E.g. the internet	
AAL Platform Plug-in Layer	Common Up-lifters & Providers, placeholder for provision of specific context data considered of common use. Common Services, placeholder for concrete service components which provide commonly needed services. Management of services is done in the Generic Platform Services Layer Common UIs, I/O channel managers reside in this building block. Common Rules, for the lower level generic platform service layer block of Automatic Situational Assistance. Common Model Extensions, pluggable components holding extra model entities.	
AAL services & applications layer	AAL services and applications that build on top of the universAAL middleware. (Created by 3rd parties)	

Table 5: universAAL Layer model

Status:

The universAAL project is in active development. The universAAL platform software is currently open to the public, in a first version. Preliminary and final documents of the project are available as well, that describe the system elaborately.

Openness:

As stated in the project’s description one of the aims of universAAL is to create an open AAL platform. It is planned to release it under a loose open source license.

Richness / Functionality:

Due to the on-going development no conclusion about the final implemented functionality and richness of services can be made. Due to the fact, that universAAL consolidates a group of AAL projects, a large number of components and services could be considered as candidates for future use: The planned architecture promises to have various facilities for the AALuis middle ware layer,e.g. hardware access, transparency of communication, and other generic platform services to be used.

Depending on the traction that the universAAL “uStore” will create, additional services can be expected in the future.

Since the software platform is not yet published, no conclusion about the concrete components can be made.

Communication:

Transparent communication between distributed services, running on network enabled nodes that have the universAAL middleware imprinted, is allowed through the middleware layer. Network enabled devices and services, that are not universAAL-aware, can be bound by the hardware abstraction service.

The architecture foresees an AAL space gateway that is used to open the local universAAL ecosystem to the outside world. This can be external services or other AAL middleware.

Facilities for user interfaces are part of the reference architecture. They are reflected in generic platform services as well as communication busses for input and output of the system. This means user interfaces need to run on nodes that are universAAL enabled.

The universAAL middleware employs a bus system for location-transparent message exchange of universAAL-enabled entities.

External entities can communicate through a hardware abstraction layer with the system.

3.5 Philips NetTV

The Philips NetTV platform consists of embedded functionalities on NetTV-enabled television sets and the infrastructure behind it. The platform allows for on-demand services which are accessible through Philips Service Portal. The actual services are deployed on servers outside the platform; the Service Portal acts as a gateway that connects the TV user with the according service, after which a direct IP connection is established.

A schematic of NetTV infrastructure can be found in Figure 10:

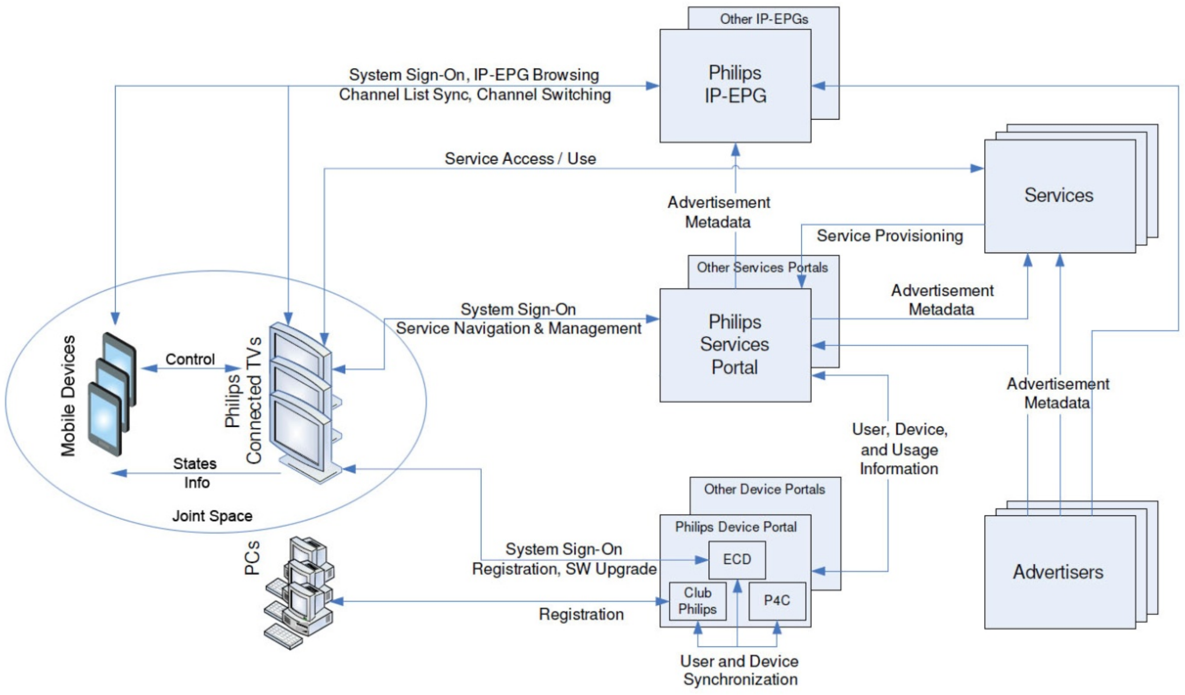


Figure 10: Philips NetTV architecture, basic overview. Services are hosted on external, 3rd party servers. The TV users connect directly to the Philips Services Portal. After selecting a service, a direct link is created with the service provider. The whole experience is integrated with the remaining TV functions (EPG).

In addition, Philips is currently working on establishing protocols for interacting with TV directly with various external devices, such as tablet computers, smartphones etc. This is based on UPnP and JointSPACE and allows for exchange and manipulation of data between devices. The goal is to make a more ubiquitous experience for the user where, for example, it would be possible to view pictures from the phone on the TV or select channels directly from a tablet computer.

3.5.1 Philips NetTV Layer model

Layer	*Description*	*AALuis use*
Television Channels / EPG	These are provided either digitally or analogously by the cable company. The EPG information allows for channel switching via a visual interface and viewing broadcasting schedule (if this is included in the broadcast by the provider)	
Service Portal	The TV interface connects directly to the Service Portal, which stores the users' information. No information is stored locally on the TV, except some basic settings. The Service Portal allows for subscription to services (via embedded App Gallery service), customizing the shortcut list and providing feedback on 3rd party services. The Service Portal also connects to advertisers, which allows for localized / targeted ad integration in the interface and EPG.	

Services	The services are hosted by their providers, on a server of their choice. Because any service in essence is a website, choices of hosting and implementation vary highly. The services are registered at Service Portal and must comply with ceHTML standards.	
JointSPACE	This is an open source project with the goal of enabling devices to interact with the TV EPG. In this way channel, and other, information between the TV and other devices can be shared and synchronized.	

Table 6: Philips NetTV Layer model

3.6 Verklizan UMO Platform

The UMO telecare platform (also called monitoring centre platform, call centre platform or alarm centre platform) is the open telecare platform designed and manufactured by Verklizan BV. The UMO is an open platform to which an enormous range of care and security equipment can be connected. This varies from the traditional personal alarm devices to advanced set-up boxes for video and telehealth applications.

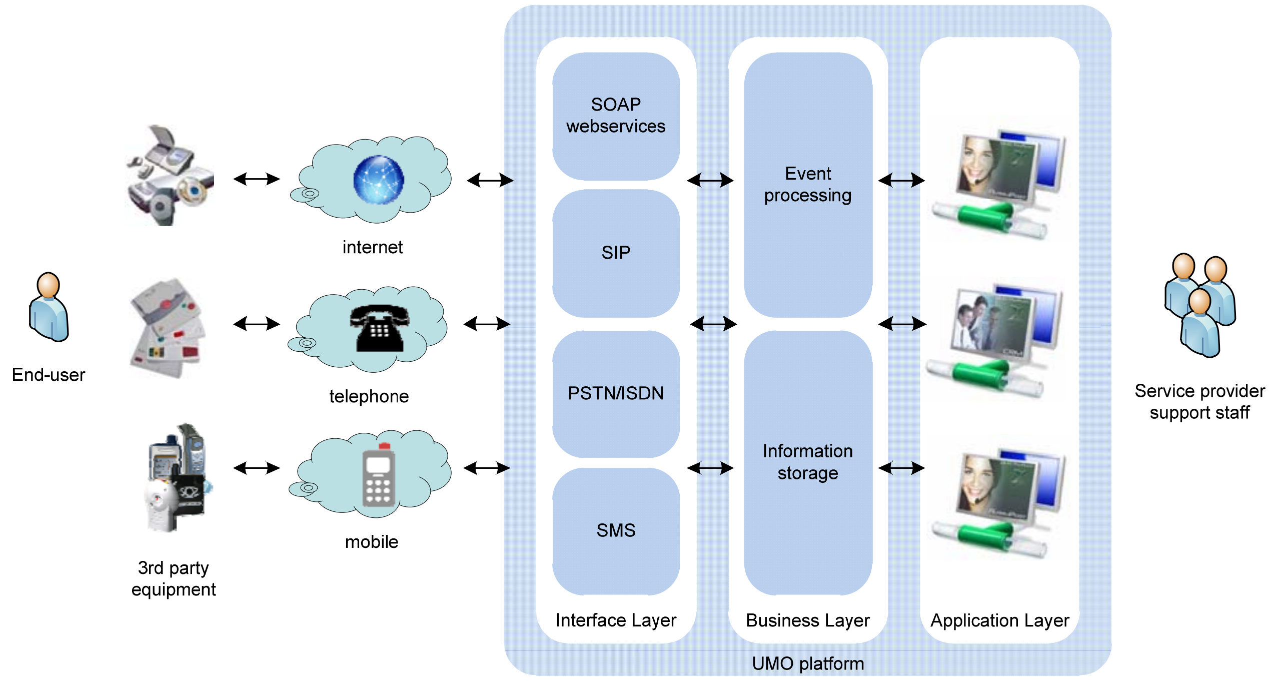
Verklizan delivers the UMO platform as a product to AAL service providers (such as the AALuis partner Hilfswerk) which use the platform to offer AAL services to end-users. So Verklizan is a platform provider and the clients of Verklizan are the service providers which deliver their services based on the platform.

In general the service providers operate as a hub between:

- (Elderly) persons in need of AAL services
- Caregiver organizations and personal carers
- Care homes & nursing homes
- Organizations installing and servicing AAL equipment
- Financing bodies

To support the service providers in their daily business, the UMO platform delivers a wide range of services for several types of users.

3.6.1 Verklizan UMO Platform Layer model



The UMO telecare platform can be divided into three architecture layers which are the interface layer, the business layer and the application layer. The interface layer provides the connection with the public communication networks, namely the telephone, mobile and internet. This layer provides protocol conversions to be able to communicate with a wide range of third party equipment. The interface layer delivers all incoming information in an uniform format to the server layer and vice versa.

The business layer provides the central processing of all incoming information delivered by the interface layer, the control of outgoing information towards the interface layer and the storage of information. In this layer information is combined and translated to business events such as social alarms and incoming calls. These business events are routed to the appropriate handler for each event. This could be an automatic handler (i.e. for automatic processing and logging) or an alarm handling application in the application layer (i.e. for manual handling by the service provider support staff).

The application layer contains the end user applications of the UMO telecare platform that enables the service provider support staff with a wide range of tasks. In the scope of AAL end-user services, the service provider support staff is supported by a call centre application which enables the handling of calls, social alarms, mobile alarms, telemonitoring, outbound contacts, screen to screen contact and video monitoring.

Other end user applications support the service provider with back office related tasks such as scheduling, financial reporting, relation management, equipment management, call centre management etc.

The platform contains no end user applications which deliver AAL services to the end-user. Those AAL end-user services are always provided to the end user by means of 3rd party telecare and security equipment.

The business model is B2B, the platform is supplied to AAL service providers. No telecare equipment is supplied with the platform. Instead, there is a close cooperation with all the manufacturers of telecare equipment, to ensure maximum compatibility with UMO and to provide complete freedom of choice to our customers. The UMO platform is open and universal: all possible telecare services could be included. The UMO business model stimulates innovation: new services can be showcased at low initial costs and prove themselves in practice

In order to provide a high reliable and available solution, all system layers are implemented with high quality components. Also the vital system components are implemented redundantly, so there is no single point of failure.

Layer	*Description*	*AALuis use*
Application Layer	The application layer delivers the UMO functionalities to the staff of the monitoring centre (i.e. the service provider) by means of a front office and a back office end user application.	-
Business Layer	The business layer provides central uniform processing, storage, control and routing of incoming and outgoing information in the UMO system.	-
Interface Layer	The interface layer translates information between the public telecommunication networks/communication protocols and the UMO platform internal communication message format for a uniform handling of information, regardless of telecommunication network or protocol implementation.	End user AAL services of the UMO platform (as described in more detail in D4.1) can be exposed to third party platforms by means of available protocol and network interface implementations. Especially open IP based communication standards such as SOAP based web services and SIP (streaming audio and video) are suitable to integrate the UMO platform with other platforms such as AALuis.

Table 7: Verklizan UMO Layer model

3.7 How is the middleware interacting with other parts

The middleware interacts with many different parts, such as services, user interfaces, devices, etc. The purpose of the middleware is to glue these components together make them interoperable. Table 8 provides an overview with which parts the AAL middlewares interact and describes how this is realized.

AAL Middleware	*Interaction with other parts*	*Description*
AMIGO	Platforms, Devices, Consumer Services, User Interfaces	
MonAMI	Services, Devices, User Interfaces	
openURC	Services, Devices, User Interfaces	
universAAL	Platforms, Devices, AAL Services,	
PhIlips NetTV	Consumer Services	
Verklizan UMO	AAL Services	

Table 8: Interaction with other parts

3.8 Interoperability & Standardization, Extendibility, Accessibility & Usability

AAL Middleware	Interoperability	Standardization	Extendibility	Accessibility	Usability	Description
AMIGO	++	+	+			Interoperability is between different technologies is realized by the service discovery and interaction middleware. Platform provide OSGi and .NET deployment frameworks
MonAMI	~	-	+			The implementation of interoperability in MonAMI is realised through the use of URC/UCH connectors and as far as visible restricted to this. The OSGi services used are thoroughly described and open, thus could be used to extend the platform. There is no information on accessibility and usability test results available.
URC / UCH	~	++	+			The URC is a standard in itself; by the addition of the UCH developments UPnP based devices can be interoperated. The URC foresees distinct points of extendibility in the architecture. Concerns of accessibility and usability are not part of the URC / UCH in itself, but have to be covered by the implementers of the system.
universAAL	++	++	++			UniversAAL shows promise to support a wide variety of hardware components. The open developments promote themselves for interoperability and extendibility. The universAAL platform is described to develop “into one consolidated, validated and standardised European open AAL platform”. The universAAL project explicitly mentions developer tools and a developer depot, to enable and facilitate the extension of the platform and services. Guidelines help in that regard. The uStore concept is promoted as a way to deploy and monetise services.
PhIlips NetTV	~	+	++			NetTV services run in the backend and are accessed via the NetTV service portal. The TV set mainly acts as user interface.
Verklizan UMO	++	++	+	~	~	IP interoperability based on implementations of the SOAP and SIP standards. Improvement of the UMO platform is a continuous process within Verklizan and therefore it is possible to extend the system if this fits the needs of Verklizan.

Table 9: Interoperability, Standardization, Extendibility, Accessibility, Usability

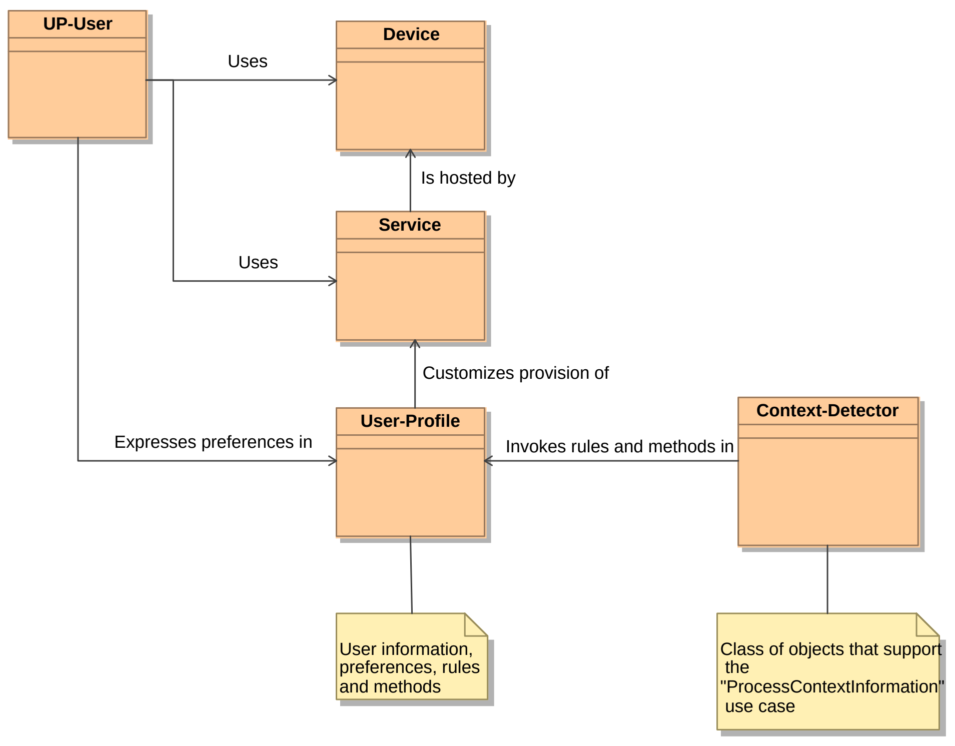
4 Area 4: Personalization and User (Interaction) Profiling

4.1 In which aspects is user profiling important on your level?

4.2 Cross Platform User Profiles WP2/3

4.2.1 ETSI – User Profile Management

The ETSI Specialist Task Force 342 on *Personalization and User Profile Management Standardization* has released various guides, technical specifications and standards dealing with user profile management [3] [4] [5]. They deal with the concept of user profiles representing information, preferences and rules used by devices or services for customization. The idea is to use the same user profile for the personalization of different services and devices as depicted in Figure 11 based on the UPM system model displayed in Figure 12. Details can be found the beforehand mentioned documents.



UP-User
Uses
Device
Is hosted by
Service
Uses
Customizes provision of
User-Profile
Context-Detector
Expresses preferences in
Invokes rules and methods in
User information, preferences, rules and methods
Class of objects that support the "ProcessContextInformation" use case

Figure 11: Class interaction model for UP/UPM based on use case model [4].

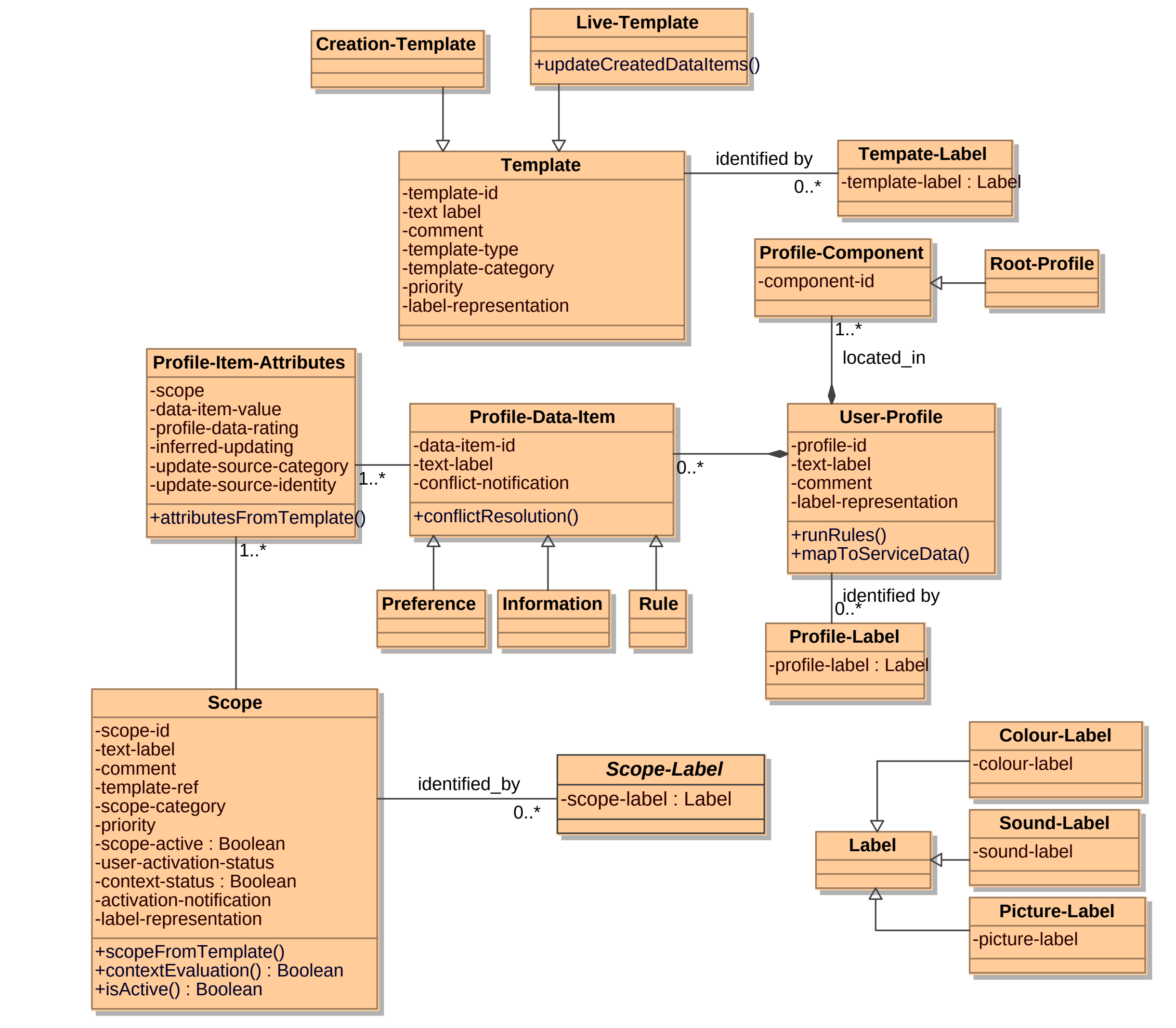


Figure 12: UPM system model [5]

4.2.2 MASP (Multi-Access Service Platform)

Blumendorf et al. present their Multi-Access Service Platform which is a runtime created for rapid prototyping, development and deployment of model-based multimodal applications [6]. Their approach is based on the following five characteristics of user interfaces: adaptivity, session management, migration, distribution and multimodality.

The concepts of adaptivity represented by a layout engine for the specification of layout constraints for rendering the user interface and migration of user interfaces between devices by means of switching between interaction channels are of interest. The information about layout is derived from different user interface models, like the Concurrent Task Tree (CTT) [7] or the interaction description. The combination of migration and adaptivity allows switching between devices with different capabilities like a mobile phone or a desktop PC [6].

4.2.3 UserML and GUMO (General User Model Ontology)

Heckmann et al. describe the user model exchange language UserML, which is based on RDF, to enable decentralized systems to communicate over user models [8]. The idea of the approach is the use of UserML [10], which can be used for modeling user model statements in a uniform way. Due to the uniform syntactical relational data structure the statements can on the one hand be represented by ontologies and on the other hand it is possible to store mass data in a database. The statements are either locally or globally stored in repositories, which can as well be shared between devices (see Figure 13). GUMO, General User Model Ontology [10], is used the basic dimensions of the user models are split into three chunks, namely auxiliaries, predicates and ranges.

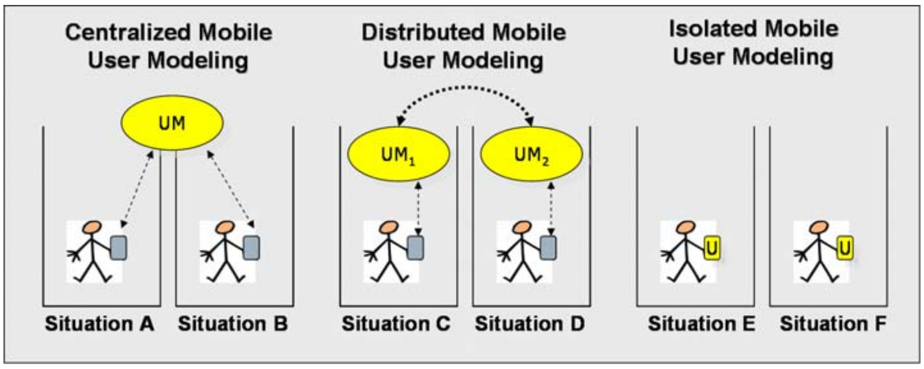

Centralized Mobile User Modeling
Distributed Mobile User Modeling
Isolated Mobile User Modeling
UM
UM1
UM2
U
U
Situation A
Situation B
Situation C
Situation D
Situation E
Situation F

Figure 13: Concept of centralized, distributed and isolated mobile user modelling [8].

4.2.4 User Preferences and Device Capabilities Profiles

Kuhnen et al. state that the success of future Next Generation Network (NGN) services depends on the ability to adapt and personalize service delivery according to the user's context as well as service and device capabilities [11]. The approach is to use two profiles: the user preferences profile and the device capabilities profiles, whereas both are based on existing standards. The user preference profile is based on the Converged IP Messaging (CPM) user preference profile concept, whereas each profile covers various personal settings, and is stored in the XML Document Management (XDM) format by the Open Mobile Alliance. Furthermore the device capabilities profile is used to store an overview of available devices and their capabilities. For the representation of the profile the Composite Capabilities / Preferences Profile (CC/PP) based User Agent Profile (UAProf) standard is used, whereas each device has its own profile. It is again based on RDF and the specification can be extended to cover all needed capabilities. The CC/PP was originally developed by the Device Independence Working Group of the Word Wide Web Consortium (W3C) [12].

4.2.5 Rule-based Adaptation Strategy

Yang et al. describe a rule-based adaptation strategy as an efficient way to transform Web contents into adapted ones that conform to the requesters' contexts [13]. The strategy is as well applicable to other content since it is a general concept for the transformation of content. Content adaptation is about transforming content based on preferences from the user, the environment and the provider. The approach adopts rule-based adaptation on context features, thus to increase the granularity of information compared to traditional methods, which are commonly based on specific environmental information and a composition of objects. The decomposition allows exact meeting of delivery context requirements posed by a heterogeneous environment. It also makes the addition of new rules possible in an easy way. This necessity, to react on context feature changes, stems from the complexity of predefining rules to meet the user's contexts [13]. The description of the context is divided into three layers representing object structure, modality and fidelity. The model is based on the CC/PP and UAProf for describing the device and complemented by accessibility and situation to describe the user. Rules can be formulated as e.g. .

4.3 Accessibility Profiles WP2

4.3.1 The Common Accessibility Profile (CAP)

Carter et al. present the idea of a Common Accessibility Profile which can be used as the basis for selecting and supporting computer-related Assistive Technologies (ATs) and focuses on the improvement of accessibility to the interactions between users and systems in a standardized manner across multiple platforms [14] [15]. The approach is standardized in the ISO/IEC 24756 standard.

The idea of CAP is to create a computable representation of accessibility profiles for users, systems and the environment. The cap can be derived from predefined profiles, and represents user's abilities and needs. When a user provides a CAP, the system establishes an accordingly accessible interaction. For providing the user's CAP to the system various media, such as USB flash drives or other flash/smart cards, or RFID tags can be used.

4.3.2 SEMA4A – Simple Emergency Alerts 4 [for] All

Malizia et al. have originally introduced as an ontology for emergency notification systems accessibility [16]. The idea is to create a common language for emergency systems but as well incorporating concepts from accessibility guidelines and interactive devices. The focus has been on creating an ontology for notifications by adhering to accessibility and usability concepts. SEMA4A is based on OWL and can be divided in three sub-ontologies [16], namely EMEDIA representing concepts related to emergency and media concepts, Wafa dealing with modelling the organization, structure and navigation of information contents, and finally AccessOnto including accessibility guidelines, user profiles and actions for users with individual abilities.

4.3.3 AccessOnto - Ontology for Accessibility Requirements Specification

AccessOnto is an ontology-based toolkit for accessibility requirements specification primarily designed to provide a repository of accessibility guidelines and a specification language intended to help SW developers to incorporate accessibility requirements in their user requirement document [17]. The main aim of AccessOne is to create a means for incorporating accessibility guidelines and related activities into the process of requirements engineering. The basic structure can be described as in Figure 14, whereas the main concepts are the user and task characteristics and the guidelines.

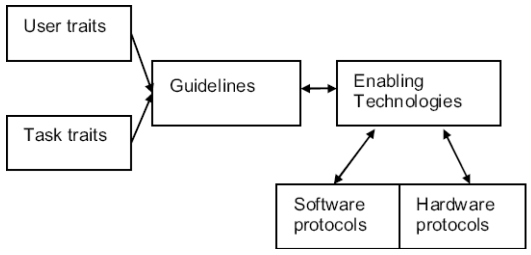


Figure 14: AccessOnto framework [17].

4.3.4 UI Adaptation System

Brunix et al. have presented an approach to combine the UIDLs, device model and user model in a UI adaptation system to ensure accessibility of data and services for individual access [18]. The information model is divided into a presentation model describing the user interface itself and the content to be presented, the user model describing the user of the service and finally the device model describing the device where the service runs. In Figure 15 the structure of the information model is depicted. The information model is described by means of the Web Ontology Language (OWL). For the presentation model User Interface Description Language (UIDL), for the user model a combination of GUMO and MPEG-21 and for the device model MPEG-21 (instead of UAProf) have been used.

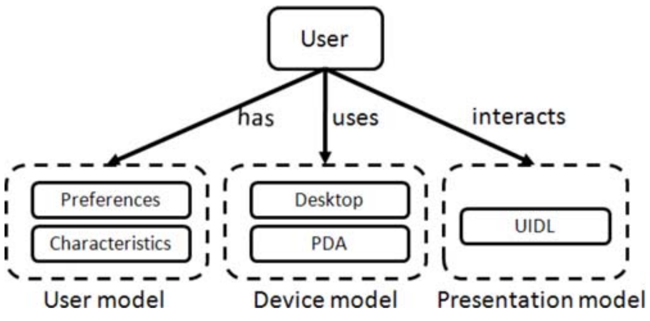


Figure 15: Structure of the information model [18].

References

[1] P. Wolf, D1.2-B AAL Reference Architecture Requirements, 2010

[2] S. Tezari, I. Gema, D1.3-B The universAAL Reference Architecture, 2010

[3] ETSI Technical Committee Human Factors, "Human Factors (HF); Personalization and User Profile Management; Architectural Framework," ETSI Technical Specification ETSI ES 202 746 V1.1.1 (2010-02), 2009.

[4] ETSI Technical Committee Human Factors, "Human Factors (HF); Personalization and User Profile Management; User Profile Preferences and Information," ETSI ETSI Standard ETSI ES 202 746 V1.1.1 (2010-02), 2010.

[5] ETSI Technical Committee Human Factors, "Human Factors (HF); User Profile Management," ETSI ETSI Guide ETSI EG 202 325 V1.1.1 (2005-10), 2005.

[6] B. Marco, S. Feuerstack, and S. Albayrak, "Multimodal Smart Home User Interfaces," in *International Workshop on Intelligent User Interfaces for Ambient Assisted Living (IUI4AAL)*, Spain, 2008.

[7] F. Paterno, *Model-Based Design and Evaluation of Interactive Applications*. London: Springer, 1999.

[8] D. Heckmann, T. Schwartz, B. Brandherm, and A. Kröner, "Decentralized User Modeling with UserML and GUMO," in *Proceedings of the Workshop on Decentralized, Agent Based and Social Approaches to User Modelling (DASUM 2005)*, 2005, pp. 61-65.

[9] D. Heckmann, "Introducing Situational Statements as an integrating Data Structure for User Modeling, Context-Awareness and Resource-Adaptive Computing," 2003.

[10] D. Heckmann, T. Schwartz, B. Brandherm, and M. von Wilamowitz-Moellendorff, "GUMO - the General User Model Ontology," in *Proceedings of the 10th International Conference on User Modeling. International Conference on User Modeling (UM), July 24-29, Edinburgh, Scotland, United Kingdom*, Edinburgh, 2005, pp. 428-432.

[11] M. Q. Kuhnen, et al., "Personalization-Based Optimization of Real-time Service Delivery in a Multi-Device Environment," in *Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE*, Budapest, 2009, pp. 1-6.

[12] W3C. (2011, Oct.) CC/PP Information Page. [Online]. http://www.w3.org/Mobile/CCPP/

[13] S. J. H. Yang and N. W. Y. Shao, "Enhancing pervasive Web accessibility with rule-based adaptation strategy," *Expert Systems with Applications*, vol. 32, no. 4, pp. 1154-1167, May 2007.

[14] J. Carter and D. Fourney, "The Common Accessibility Profile," The University of Saskatchewan - Department of Computer Science, Saskatoon, Canada, Technical Report, 2004.

[15] D. Fourney and J. Carter, "A Standard Method of Profiling the Accessibility Needs of Computer Users with Vision and Hearing Impairments," in *Conference & Workshop on Assistive Technologies for People with Vision & Hearing Impairments, technology for Inclusion, CVHI 2006*, 2006, pp. 138-142.

[16] A. Malizia, T. Onorati, P. Diaz, I. Aedo, and F. Astorga-Paliza, "SEMA4A: An ontology for emergency notification systems accessibility," *Expert Systems with Applications*, vol. 37, pp. 3380-3391, 2010.

[17] K. R. Masuwa-Morgan, "Introducing AccessOnto: Ontology for Accessibility Requirements Specification," in *First International Workshop on Ontologies in Interactive Systems*, 2008, pp. 33-38.

[18] C. Bruninx, C. Raymaekers, K. Luyten, and K. Coninx, "Runtime Personalization of Multi-Device User Interfaces: Enhanced Accessibility for Media Consumption in Heterogeneous Environments by User Interface Adaptation," in *Second International Workshop on Semantic Media Adaptation and Personalization*, 2007, pp. 62-67.

[19] G. Zimmermann, URC Technical Primer 1.0 (DRAFT), http://myurc.org/TR/urc-tech-primer1.0/ (14.10.2011)

[20] G. Zimmermann, G. Vanderheiden, R. Charles, Universal Control Hub & Task-Based User Interfaces http://myurc.org/publications/2006-Univ-Ctrl-Hub.php (14.10.2011)

[21] E. Gómez-Martínez, J. Mersegeur (2010), „Performance Modeling and Analysis of the Universal Control Hub“ in EPEW'10 Proceedings of the 7th European performance engineering conference on Computer performance engineering, Springer-Verlag, Berlin, p 160-174, ISBN: 3-642-15783-1 978-3-642-15783-7

[22] ISO/IEC (2008) Information Technology - User Interfaces – Universal remote console – Part 2: User interface socket description, ISO/IEC 24752-2:2008, Geneva

[23] ISO/IEC (2008) Information Technology - User Interfaces – Universal remote console – Part 3: Presentation template, ISO/IEC 24752-2:2008, Geneva

[24] Swedish Institute of Assistive Technology, MonAMI Information http://www.monami.info (14.10.2011)

[25] P. Steiner et al (2011), D23.2 Annex MonAMI Architecture description

[26] MonAMI Project Consortium (2011), MonAMI Core Services, User Manual

[27] universAAL Project Consortium (2011), universAAL.org, http://universaal.org/index.php?option=com_content&view=article&id=7&Itemid=6

[28] S. Tazari, O. Höftberger, Z. Owda, D2.2-A universAAL Generic Platform Services, AAL platform services and ontology artefacts

[29] AMIGO project website, http://www.hitech-projects.com/euprojects/amigo/software.htm (22.11.2011)