IST Amigo Project

Deliverable D2.2

State of the art analysis including assessment of system architectures for ambient intelligence

Public

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

Deliverable Number	:	D2.2
Title of Deliverable	:	State of the art analysis including assessment of system architectures for ambient intelligence
Nature of Deliverable	:	Public
Internal Document Number	:	Amigo_D2.2_WP2_final.doc
Contractual Delivery Date	:	28 February 2005
Actual Delivery Date	:	11 April 2005
Contributing WPs	:	WP2
Author(s)	:	**France Telecom:** Fano Ramparany, Jérôme Pierson, Thibaud Flury, Gilles Privat, Anne Gérodolle **Ikerlan:** Herrasti Natividad **Inria:** Frédéric Le Mouël, Laurent Réveillère, Nikolaoas Georgantas, Sonia Ben Mokhtar, Ferda Tartanoglu, Valérie Issarny, Christophe Cerisara. **Knowledge:** Basilis Kladis **Microsoft:** Mark Gilbert, Ron Mevissen **Philips:** Bram van der Wal, Harmke de Groot, Michael van Hartskamp, Peter van der Stok **Stichting Telematica Instituut:** Henk Eertink, Remco Poortinga, Tom Broens **Telefonica:** Sara Carro Martinez, Javier Arribas, José María Miranda **Fraunhofer IMS:** Gerd vom Bögel **VTT**: Päivi Kallio, Jiehan Zhou, Julia Kantorovitch

Abstract

This report presents state-of-the-art system architectures that serve as technological building blocks and background technologies for the Amigo system, which aims to enable ambient intelligence in the networked home environment. The Amigo project specifically aims to solve the main technological issues that endanger the usability of a networked home system in which traditionally separated domains (i.e., home automation, personal computing, consumer electronics and mobile communications) need to be effectively merged. The Amigo project then investigates solutions for the seamless integration and improvement of services from the four domains, offering home users easy, intelligent and meaningful interfaces and services.

This report presents:

The technological building blocks for the networked nodes of the home environment including wireless and stationary computing nodes, home networking technologies, IP-based protocols, real-time protocols, and operating systems.

Most popular existing service-oriented middleware architectures like OSGi, UPnP, Web services and service discovery protocols, which enable interoperability among heterogeneous applications deployed on the heterogeneous devices of the networked home environment;

Middleware support for security and privacy, QoS, and accounting and billing;

Intelligent user services, including context management, multimodal user interfaces, and user modelling and profiling, which make home environment much more attractive to the user;

Software system architectures aimed at ambient intelligence proposed within projects such as MIT Oxygen, IST Ozone, and ITEA Ambience.

Keyword list

Ambient intelligence, ambient system architecture, home system, service-oriented middleware, service discovery protocols, QoS, security, privacy, intelligent services, context awareness, multimodal interfaces, modelling, profiling.

Table of contents

Figures

Tables

List of Abbreviations

3GPP	3rd Generation Partnership Project
AAA	Authentication, Authorization and Accounting
AAAARCH	Authentication Authorization Accounting ARCHitecture Research Group
ABR	Associativity Based Routing
ACL	Access Control List
AES	Advanced Encryption Standard
AODV	Ad-hoc on Demand Distance Vector
AmI	Ambient Intelligence
AP	Access Point
API	Application Programming Interface
AWML	Augmented World Modelling Language
B2B	Business to Business
B2C	Business to Consumer
BPML	Business Process Modelling Language
BPEL	Business Process Execution Language
CDR	Charging Data Record (also, Call Detail Record)
CD	Compact Disk
CDMA	Code-Division Multiple Access
CE	Consumer Electronics
CIP	Cellular IP
CF	Compact Flash
CFS	Cooperative File Systems
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CSCW	Computer Supported Cooperative Work
DA	Directory Agent
DAML	DARPA Agent Markup Languages
DAML-ONT	DAML Ontology
DHCP	Dynamic Host Configuration Protocol
DHW	Domestic Hot Water
DiffServ	Differentiated Services
DLNA	Digital Living Network Alliance
DMP	Digital Media Player
DMS	Digital Media Server
DNS	Dynamic Name Service
DRM	Digital Rights Management
DSDV	Destination-Sequenced Distance-Vector

DTCP	Digital Transmission Content Protection
DTLA	Digital Transmission License Administrator
DSL	Digital Subscriber Line
DVD	Digital Versatile Disc
DVR	Digital Video Recorder
E21s	Enviro21s
EDGE	Enhanced Data GSM Environment
EMMA	Extensible multi-modal annotations
EPG	Electronic Program Guide
EGNOS	European Geostationary Navigation Overlay Service
ESD	Electronic Software Distribution
FCM	Functional Control Module
H21s	Handy21s
HCI	Human Computer Interaction
HDTV	High-Definition Television
NMIP	Hierarchical Mobile IP
HTTP	HyperText Transfer Protocol
HTTPMU	HTTP multicast over UDP
HTTPU	HTTP unicast
HW	Hardware
ICC	Integrated Circuit Card
IDE	Integrated Drive Electronics
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
FCC	Federal Communications Commission
IDEA	International Data Encryption Algorithm
IHDN	In-Home Digital Networks
INS	Intentional Naming System
IMS	IP Multimedia Subsystem
IntServ	Integrated Services
IP	Internet protocol
IPDR	Internet Protocol Detail Record
IRTF	International Research Task Force
ISP	Internet Service Provider
IST	Information Society Technologies
IT	Information Technology
JERRFREE	Java Embedded Framework FREE
JVM	Java Virtual Machine
KDC	Key Distribution Centre
KDE	K Desktop Environment

LAN	Local Area Network
LCD	Liquid Crystal Display
LGPL	Lesser General Public License
LOS	Line of Sight
MANET	Mobile Ad-hoc NETworks
MEG	Mobile Expert Group
MIT	Massachusetts Institute of Technology
MIP	Mobile IP
MMS	Multimedia Message Service
MPEG	Moving Picture Experts Group
N21s	Networks
NDM-U	Network Data Management – Usage
O2S	Adaptable software systems
OIL	Ontology Inference Layer
OLSR	Optimize Link State Routing
OS	Operating System
OSGi	Open Services Gateway Initiative
OWL	Ontology Web Language
PAN	Personal Area Network
PC	Personal Computer
PDA	Personal Digital Assistant
PER	Packet Error Rate
PML	Physical Markup Language
PMP	Personal Media Player
PSNR	Peak Signal to Noise Ratio
PVA	Physical-Virtual Artefacts
PVR	Personal Video Recorder
GPL	General Public License
GDSP	Gatespace's Distributed Service Platform
GENA	Generic Event Notification Architecture
GLONAS	Global Navigation System
GML	Geography Markup Language
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
QoS	Quality of Service
RAM	Random Access Memory
RAW	Raw Architecture Workstation
RBAC	Role Based Access Control
RSA	Rivest-Shamir-Adleman encryption algorithm
RDF	Resource Description Framework

RDFS	Resource Description Framework Schema
RF	Radio Frequency
RMI	Remote Method Invocation
RSVP	Resource Reservation Protocol
RTFM	Real-Time Traffic Flow Measurement architecture
RTP	Real-time Transport Protocol
Scale	Software-Controlled Architectures for Low Energy
SA	Service Agent
SD	Secure Digital
SDTV	Standard Definition TV
SESAME	Secure European System for Applications in a Multi-vendor Environment
SMS	Short Message Service
SSL	Secure Sockets Layer
SFS	Self-Certifying
SIP	Session Initiation Protocol
SLP	Service Location Protocol
SNMP	Simple Network Management Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSDP	Simple Service Discovery Protocol
SGADK	Service Gateway Application Development Kit
SVG	Scalable Vector Graphics
SW	Software
TCP	Transport Control Protocol
TBRPF	Topology Dissemination Based on Reverse-Path Forwarding
TripleDES	Triple Data Encryption Standard
TLS	Transport Layer Security
TORA	Temporally Ordered Routing Algorithm
TRIP	Telephony Routing over IP
TV	Television
UA	User Agent
UBR	Universal Business Registry
UDP	User Datagram Protocol
UDDI	Universal Description, Discovery and Integration
UMTS	Universal Mobile Telecommunications System
UPnP	Universal Plug and Play
URL	Uniform Resource Locator
USB	Universal Serial Bus
UWB	Ultra Wide Band
VCR	Video Recorder

VoI	Voice over Internet
WAN	Wide Area Network
WASP	Web Architectures for Services Platforms
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WS	Web Services
WS-CDL	Web Service Choreography Description Language
WSDL	Web Services Description Language
WWAN	Wireless Wide Area Network
XCAP	XML Configuration Access Protocol
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
XPATH	XML Path Language

1 Introduction

Ambient intelligence emphasises greater user-friendliness, focusing on a “smart way” of using communication technology to make life simpler, more enjoyable, interesting and empowered. Ambient intelligence refers to electronic environments that are sensitive and responsive to the presence of people [AM03] [A01] [EA03] [R03] [W93] and it can be defined as the merger of two important visions: **ubiquitous computing** [RN96] [W93] and **social user interfaces** [L03]. Ambient intelligence builds on advanced networking technologies, which allow robust, ad-hoc networks to be formed by a broad range of mobile devices and other objects, i.e., ubiquitous computing [RN96]. By adding adaptive user-system interaction methods based on new insights into the way people like to interact with computing devices (i.e. social user interfaces), digital environments can be created, which improve the quality of life of people by acting on their behalf.

Key characteristics of ambient environments are: ubiquity, awareness, intelligence, and natural interaction. **Ubiquity** refers to a situation in which we are surrounded by a multitude of interconnected embedded systems, which are invisible and moved into the background of our environment. **Awareness** refers to the ability of the system to locate and recognize objects and people. **Intelligence** refers to the fact that the digital surrounding is able to analyse the context, to adapt itself to the people that live in it, to learn from their behaviour, and eventually to recognize as well as show emotion. **Natural interaction** means advanced modalities like speech-, gesture- and object-recognition, which will allow more natural communication with the digital environment than is possible today [ISTAG01]. Ambient intelligence involves the convergence of ambient computing, ambient communication and ambient interaction areas. **Ambient computing** contributes to the development of various ad-hoc networking capabilities that exploit highly portable or very-low-cost computing devices. **Ambient communication** contributes to the development of ubiquitous communication capabilities between end-users and ambient computing devices through various ad-hoc or wireless networks. **Ambient interaction** contributes to the development of human-centric user interfaces that allow people to interact with their environment in a natural way.

This report aims to provide an overview of state-of-the-art technological developments towards enabling ambient intelligence for the networked home environment, where the traditionally separated domains of home automation, consumer electronics (CE), mobile communications, and personal computing (PC) are merged and given intelligence, thus offering home users easy, intelligent and meaningful interfaces to handle information and services ubiquitously. Tremendous efforts have been and are currently being poured into developing concepts for enabling such an ambient intelligence vision. As an example MIT's Oxygen project can be mentioned for bringing abundant computation and communication into people's lives, as pervasive and free as air [D99], and CMU's Aura for distraction-free ubiquitous computing [SG02]. Most of the current approaches aim to increase people's professional productivity [D01].

Amigo’s ambient intelligence differs from these proposals by focusing on bringing a new kind of interaction with ambient computing technology into our homes and personal domains, thus promoting our experiences and lives. The Amigo project in particular builds on past projects to which the consortium partners actively contributed and were specifically focused on such issues, e.g., the ITEA Ambience and IST Ozone projects.

As enabling the ambient intelligence and related pervasive computing vision has led to and is still the subject of extensive scientific and technological developments, this report does not pretend to provide an exhaustive overview of all the latest results that are relevant to the definition of system architectures enabling ambient intelligence. Instead, the report focuses on technological developments that will be used as building blocks and/or background for the definition of the Amigo system architecture, highlighting architectural issues that have yet to be addressed towards meeting Amigo objectives.

This report is structured as follows.

Chapter 2 presents platform and infrastructure elements including wireless and stationary computing nodes, home networking technologies, IP-based protocols, real-time protocols, and operating systems. In the context of Amigo, these technologies will be reused, possibly exploiting the latest developments available during the course of the project.

The Service-Oriented Architecture (SOA) approach appears to be a convenient architectural style towards meeting one of the key objectives of the Amigo project, that is, to enable interoperability among heterogeneous applications deployed on the heterogeneous devices of the networked home environment. The most popular existing software service-oriented middleware architectures like OSGi, UPnP, Web services including service composition and semantic modelling, service discovery protocols, middleware support for security and privacy, QoS, and accounting and billing are overviewed in Chapter 3.

Chapter 4 is focused on intelligent user services, including context management, multimodal user interfaces, and user modelling and profiling, which make home environment much more attractive to the user.

Software system architectures aimed at ambient intelligence proposed within such projects as MIT Oxygen, IST Ozone, and ITEA Ambience and referenced in Amigo are surveyed in Chapter 5.

Chapter 6 concludes the document.

References:

[AHS01] Aarts, E., Harwig, R., Schuurmans, M., “Ambient Intelligence”, in Denning, P.J. (Ed) The Invisible Future, ACM Press. pp. 235-250, 2001.

[AM03] Aarts, E.H.L., Marzano S., “The New Everyday, Views on Ambient Intelligence”, 010 Publishers, Rotterdam, The Netherlands. ISBN 90-6450-502-0, 2003.

[CDK01] Coulouris G., Dollimore J., Kindberg T., “Distributed Systems: Concepts and Design”, Edition 3 Addison-Wesley, Pearson Education, 2001.

[R03] De Ruyter, B., “365 Days” HomeLab, Royal Philips Electronics, http://www.research.philips.com/technologies/misc/homelab/downloads/homelab_365.pdf

[D99] Dertouzos, M.L., “The Future of Computing”, scientific American 281 (2), pp. 52-55, 1999.

[D01] Dertouzos, M.L., “The Unfinished Revolution: How to Make Technology Work for Us-Instead of the Other Way Around”, New York, Harpercollins, 2001.

[D04] Duley, C., “GNVQ Advanced IT”, 2004, http://www.btinternet.com/~C.J.Duley/gengloss.htm

[EA02] Eggen, J.H., Aarts, E.H.L. (Eds), “Ambient Intelligence in HomeLab”, Royal Philips Electronics, ISBN 90-74445-55-1, 2002.

[ISTAG01] ISTAG, “Scenarios for Ambient Intelligence in 2010; Final Report”, Feb 2001, ftp://ftp.cordis.lu/pub/ist/docs/istagscenarios2010.pdf

[RN96] Reeves, B., Nass, C., “The Media Equation”, Cambridge University Press, 1996.

[SG02] Sousa, J.P, Garlan, D., "Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments", 3rd Working IEEE/IFIP Conference on Software Architecture, 2002.

[L03] Van Loenen, E.J., “On the Role of Graspable Objects in the Ambient Intelligence Paradigm", Smart Objects Conference (SoC), Grenoble, 15-17 May 2003.

[W93] Weiser, M., “Hot Topics: Ubiquitous Computing”, IEEE Computer, October 1993.

2 Platform and infrastructure elements

The technological building blocks for the networked nodes of the home environment decompose into networked devices (or nodes) (§2.1), the networking technologies (or networks) (§2.2) and operating systems (§2.3), for which we concentrate on the technologies that are the most used today. In the context of Amigo, these technologies will be reused as is, possibly exploiting the latest developments available during the course of the project. However, it is acknowledged that further developments are in general needed to meet the overall requirements of ambient intelligence.

2.1 Nodes

2.1.1 PDAs and Smart Phones

Personal Digital Assistants (PDAs) and Smart Phones are becoming more and more common everyday. There are millions of modern PDAs in the world, and even more Smart Phones. The usage of both is expected to increase, and the smart phone is expected to become as common as the normal mobile phone is today (over a billion).

Although they have different origins and primary purposes, the technology and software included in each is increasingly similar. The PDA was originally used as a personal productivity device to hold personal data such as contact information, schedule information, and short notes. The PDA has evolved and is now most often connected to the network with wireless LAN or wireless WAN radios (often both). The applications have also evolved into a personal set of office applications with e-mail being a primary application because of the connectivity. On top of this, many PDAs also have pure voice capabilities so they can be used as a regular mobile phone on mobile networks. These devices often have CPUs in the hundreds of megahertz and memory between 32 and 100 MBs of RAM. Because of the continually decreasing cost of storage, these devices often have a compact flash or SD memory reader, which enables the device to have non-volatile storage in the low gigabytes. The PDA often has little or no buttons on the outside, and primarily uses touch screens and pen input for navigation and data entry. The PDA also has a screen that is typically rectangular with a physical size of 8cm x 4cm and resolutions between 320x240 and 640x480 pixels.

The Smart Phone has a similar set of applications as a PDA, including simple office type applications and personal information applications, however its primary purpose is to be a mobile phone. Smart Phones always have a WWAN radio and are increasingly coming with WLAN radios as well. The hardware available on Smart Phones is similar that of PDAs but generally they have a slower CPU to increase battery life (an important factor on both devices but currently considered more critical on the Smart Phone) and input limited to a touch pad of numbers and/or characters. Smart Phones will generally have a physically smaller screen and lower resolution (150x300 pixels).

Both of these devices now have CPUs around or in the hundred megahertz area with megabytes of RAM and expansion for storage into the gigabytes. The operating systems are multi-threaded and support 3rd party applications being easily developed and installed on them. The primary difference between the two devices in relation to Amigo is the user input and capabilities. Owners of these devices generally carry them at almost all times. The PDA has a richer user input and display but it is hard to input data on while moving (because of the touch screen). As a trade off, the Smart Phone is often a smaller and more rugged device.

2.1.2 PC

The Personal Computer (PC) is one of the most common and powerful computing devices in the home. Most personal computers now contain processors over 1 gigahertz, a large amount of storage (between gigabytes and a terabyte of storage), a rich graphical user interface (high resolution screen, keyboard and mouse input), and a number of networking technologies (Ethernet, IEEE 1394, 802.11x and USB). All these technologies combine to give a very versatile node in the home network. PCs can run a number of operating systems. Windows is the most common, followed by Mac OS and Linux. Although some run on different hardware, their overall power and capabilities are fairly similar. The PC is currently being used for many things and applications are continually growing.

Some newer examples are:

- **Media Manager and Server**: Many people use PCs to create or capture media content like music and video. With the inclusion of DVD drives in most PCs, they are also getting used to store and playback full-length movies. PCs are now getting used to also set-up play lists and organize content for other devices to use because of the PC's richer interface. For example, there are stereo components that play digital music and use play lists that are stored and configured on the PC.
- **Communication Device**: E-mail and instant messaging are both popular forms of communication, which the PC is the primary portal for. Beyond traditional person-to-person communication, these mediums are also being used as alerts and reminders for news, travel information, and bill paying.

References:

The usability of everyday technology: emerging and fading opportunities http://portal.acm.org/citation.cfm?id=513667&coll=portal&dl=ACM&CFID=8745503&CFTOKEN=37406551

Windows Media Center Edition Information and Scenarios http://www.microsoft.com/windowsxp/mediacenter/evaluation/default.asp

Windows XP Home Edition Information http://www.microsoft.com/WindowsXP/home/default.asp

Microsoft Developers Network http://msdn.microsoft.com/

2.1.3 TV and set-top box

The TV is seen as a good entry point in the domestic entertainment world, while the home may be viewed as a place to relax by its inhabitants. Big screens thrill in the family room and a large number of homes have products like home theatre and digital satellite systems that enhance the TV viewing experience.

With home theatre systems, plasma and LCD high-definition TVs, and digital video recorders moving into the living room, the way people watch TV will be radically different in five years' time.

The most talked-about technologies today are digital and personal video recorders (DVR and PVR). These set-top boxes allow people to record, store, play, pause and forward TV programs whenever they want. Essentially, they allow a much more personalized TV experience.

These technologies are also being built-in to high-definition TV sets, which are big business in Japan and the US, but slower to take off in Europe because of the lack of high-definition programming.

Not only can people forward through advertisements, they can also forget about abiding by network and channel schedules, putting together their own a-la-carte entertainment.

In addition to personalized content viewing, other TV experiences like ambient lighting in the TV-set can extend the viewing experience. TV may extend content in the interior of the room so that the viewer gets more and more involved in the viewing experience.

High-definition TV (HTDV) will require high quality of service for streaming applications like watching a movie on TV. This HDTV content can be streamed from outside the home (by an external or internal set-top box) or from a server within the home.

When watching HDTV content on an HDTV set, it might be desirable to watch that content on a portable media player. Therefore the Amigo system needs to be able to transcode the HDTV content to lower bandwidths, support multiple codecs to display content and distribute it over the network.

Local storage is an area in which the application and software are foreseen to grow exponentially. Currently a server in the home for storage is most likely a PC. Also the implementation of 'set-top box' types of applications inside a TV will change the interface of TVs in a home network. The TV will become more intelligent and easier to communicate with, so the TV can become a central part of the home.

Connectivity between the TV and portable media like camcorders, Personal Media Players (PMPs) and photo cameras adds new interaction types to the TV besides only standard TV viewing. Quick media shows can be given from holiday pictures or videos by easily connecting (no annoying configuration anymore) portable media hardware to the TV and browsing through the content.

To further integrate the TV with daily routines in the home, it needs to be connected to the home network and be able to deliver different kinds of information (movies, EPG, Internet browsing, gaming, video conferencing, message etc) to the user. This requires a TV with intelligent, integrated functions that provide the previously mentioned services.

The Amigo project should investigate a novel, highly powerful system architecture, possibly including programmable and reconfigurable hardware components, to support the typical Amigo complex applications mixing real time processing and streaming oriented types of behaviour and processing that provide functionalities:

- To process typical Amigo related data such as video, audio and graphics. It is usually the goal to provide the user with the highest possible quality data given the restrictions of the terminal that the user is using, the bandwidth restrictions of the network and other restrictions such as preferences and costs.
- To support simultaneous stream accesses and routing operations.
- To find solutions for the end-to-end quality of service for video streaming.
- To improve viewing experiences in the Amigo home.
- To make the TV a key element in the Amigo home for information retrieval and entertainment

2.1.4 Game console

Game consoles often belong with the highest processing powered devices in a home. They are designed to deliver guaranteed and maximum performance for demanding real-time applications (e.g. 3D rendering, surround sound generation, virtual reality environments etc.). Game consoles provide their own platform (e.g. Sony Playstation, Microsoft XBox, Nintendo GameCube etc.) which is fundamentally different from other computer platforms:

- Application developers can get access to software development kits and tools for (game) development but this often requires licensing and approval from the manufacturers.
- Game consoles do not provide good facilities to permanently store information. Storage cards can sometimes be purchases as accessories but default permanent storage is limited compared other existing computer platforms.
- Applications are not (and can not be) installed but run from a medium (CD, DVD, memory card). As a consequence, a game console can often run only a single application at a time.
- The platform (HW and SW) cannot be upgraded. The game console is a closed device built to deliver a guaranteed (and as a disadvantage also a maximum) performance that cannot be upgraded or extended.
- Limited support for external interfaces. External interfaces are often limited or vendor specific (e.g. game controllers)

As a result, application development for game consoles concentrates on stateless, short lived entertainment services like games, music, video playback and Internet access.

Occasionally game consoles allow external hardware to be connected to improve the entertainment experience:

- Remote control. This is mainly used for music and video playback.
- WLAN and LAN access. To interconnect multiple game consoles (e.g. for multiplayer gaming) or connect a game console to the Internet.

2.1.5 Domestic appliances

Household appliances or domestic appliances (the current ones and the future ones of an installation of ambient intelligence) must be considered like a set of benefits and services obtained by means of the application of diverse technologies.

They can be divided into three groups when considering them as individualized products: White goods appliances, comfort systems and small electric appliances. On the other hand, considering them as a whole, they are treated, among many other devices at home, under the concept of “Domotic”.

White goods appliances are fridges, dishwashers, hobs, extractors, freezers, ovens, cookers, washing machines, and microwaves. Each of them is the result of the application of specific technologies to obtain the functionality that they offer (e.g. washing technologies such as detergents, agitation etc.; technologies of heat generation such as combustion, halogen lamps, induction etc.; technologies of cold generation; mechanical technologies such as anti-vibration, structural, elimination of acoustic noises; and environmental technologies such as energy, recyclability, materials, eco-design etc.). It is also common that the high range of these products is equipped with:

- Electronic embedded control systems
- Elements of visualization for the user interfaces, even touch-screens
- Different sensors to favour the automation of its processes
- Some capacities of communication and services (Domotic) to form networks of household appliances offering many improvements in the interface as well as new features (security, power saving, tele-maintenance, version update etc.).

Regarding software, the household appliances equipped with embedded systems are controlled by programs closely related to the hardware. This software can therefore be defined as firmware.

The products integrated under **comfort systems** are divided into three groups: heating (wall-mounted gas boilers, electric storage heaters), air-conditioning (air-conditioning units), and domestic hot water (DHW: wall-mounted gas boilers, gas water heaters, electric water heaters). As in the previous case, each device is the result of applying specific technologies to obtain the functionality that they offer (combustion technologies, distribution technologies, and environmental technologies such as energy, recyclability, materials, eco-design etc.). It is also common that the high range of these products is equipped with the same extensions as the white goods appliances. Their software can also be defined as firmware due to the equipment with embedded systems.

Small electric appliances form a set of innumerable small devices dedicated to very specific and heterogeneous tasks (e.g. pressure cookers, coffee pots, toasters, mixers, hair dryers, deep fryers, small electric heaters, radiators etc.). They are characterized by their simplicity and low cost, with a few exceptions that have electronic embedded controls, communication systems, and sensors.

Domotic realizes the state before the ambient intelligence state. It can be defined as a set of technological systems which equip the domestic products at home (e.g. household appliances, TVs, DVDs, comfort, illumination, security etc.) with new features in security, communication, power management, comfort, and user interface. The following technologies are related to it: software (operating systems, middleware, specific applications), embedded control systems (processing hardware, low power consumption), sensors, interface systems (multimodal), and communication systems (protocols, standards, physical mediums, interoperability initiatives).

In a domestic installation based on the concepts of ambient intelligence, the presence of white household appliances is crucial. These elements are in charge of doing the most laborious tasks in the home and, therefore, with the purpose of reducing the dedication of the users to these tasks. It is vital to improve their performance and their interface by means of the inclusion of technology.

In the Amigo domain, the knowledge reported by the different internal sensors of the household appliances (e.g. different states, phases, sub-phases etc.) must be transmitted to the diverse distributed intelligences (e.g. property of the devices, of the kitchen, of the global house etc.), so that the functional automation degree of these devices increases. It is also necessary to incorporate new methodologies of multimodal interfaces, specially based on conversational control (voice) and networks of displays, with the purpose of offering new benefits of ubiquitous control and monitoring, from the interior as well as from the exterior of the house. This last concept, the ubiquity of the control, will have to be obtained by incorporating the communication capacities that guarantee interoperability with the rest of the systems and the household appliances. Finally, embedded control systems must be developed to obtain new household appliances that offer this set of new features, individually (in a smaller manner) as well as in the network (totality). Everything is focused on obtaining final products. The perception that the market has of these elements is that they are cheap products, very adapted to a function that is rarely appreciated by the users.

It is difficult to conceive that this perception changes, because the market would not assume a price rising to improve their functionality and interface by means of the inclusion of technology. The only job of these devices, in extremely automated surroundings, would drive them by means of an external actuator (on, off, programming). The exception to this seems to be, among some others, the pressure cooker (whose functionality and security can be improved by means of the inclusion of sensors, communications and an embedded controller, so that it would be a product accepted by the market) and a very sophisticated vacuum cleaner.

In the ambient intelligence field, the research issues related to the domestic appliances domain are numerous, including:

- The new functionalities of each electric appliance working standalone and being part of the whole. These new features shall emerge as a consequence of the integration of domestic appliances in an ambient intelligence context: ubiquity in all the actions of control and monitoring, prediction of functionality according to user preferences and circumstances, reception of services from outside in real-time, natural and conversational relation etc.
- Internal sensorization of devices with the purpose of equipping them with more and better information about the diverse intelligences and users.
- Sensorization of external variables important for the control, monitoring and functionality of the household appliances: voice, presence of users by different methodologies, environmental and climatic magnitudes related to comfort etc.
- Software (operating systems, middleware, specific applications).
- Multimodal interfaces (voice, visualization networks).
- Communications that guarantee the interoperability between the different products that are complemented to offer concrete functionalities.
- Environmental technologies: energy, recyclability, materials, and eco-design.

References:

EU- Project InHomNet http://www.ict.tuwien.ac.at/ieee1394/projekte/inhomnet-en.html

Project ITEA ROBOCOP http://www.extra.research.philips.com/euprojects/robocop/

Project ITEA Space4U http://www.extra.research.philips.com/euprojects/space4u/

2.1.6 Smartcards[1]

Smart cards, also called Integrated Circuit Cards (ICCs), are replacing existing magnetic-stripe cards (such as credit cards) in many applications. Instead of storing a few hundred bytes of data on the magnetic stripe, the card contains a microprocessor and up to some ten Kbytes of memory. Thus, a smart card is actually a small, programmable computer system that is extremely powerful. Smart cards can provide greatly increased security, the ability to conduct transactions off-line, and the ability to install multiple applications on a single card. Although they are not yet widely known in the U.S., they have been extensively used in Europe and Asia since the early 1990s [C98]. The prognosis for the European market shows a growth from 794.6 million pieces in 1999 (60% of world market) to 2050 million pieces in 2006 [F00].

Several pilot programs tested smart cards, from single financial applications, electronic food stamps, a complete medical record, a doctor's appointment schedule up to the combination of several services. After the switch to an electronic core many extensions were introduced. One is the integration of security mechanisms. The security of access to a card’s content rises significantly with crypto-algorithms also realised in hardware. A recent improvement to the smart card is a contactless communication interface, which allows easier handling (invariant orientation) and gives higher reliability (ESD insensitive) [SCA03]. The contactless communication is realised by an inductive coupling, suitable for an energy and data transmission of a few centimetres. To solve the demand for higher flexibility of the supported functionality, new concepts for the download of software into the cards were invented.

[1] Text taken from Amigo SOTA, written by Thompson, updated by Philips CE – Innovation Lab

The most promising approach is the implementation of a Java Virtual Machine. Therefore, in the near future, native Java code will be executable on a smart card [BBEHOW04].

Up to now, cards are frequently used for single applications, e.g., in finance and access control. The merge of different applications into one card is extremely difficult because of the hard security and handling requirements within the logistic processes of the card suppliers. New standards and functions are able to overcome these drawbacks in principle. In general, the new generation of smart cards opens a huge variety of new applications in the field of ambient intelligence. An essential contribution of the success of smart cards in the future can bring the link to the Internet with its possibilities of data-exchange and reloadable functions. In some applications competition to smart phones appears. A smart card may be used as a carrier for data but also as a carrier for applications. When smart cards have enough processing power they are also able to execute special software, received from other components in a smart environment.

References:

[BBEHOW04] Baentsch, M., Buhler, P., Eirich, T., Hoering, F., Oestreicher, M., Weigold, T., ”Fact sheet on IBM's JCOP smart card operating system”, IBM Zurich Research Laboratory, http://www.zurich.ibm.com/csc/infosec/java_factsheet.html , November, 2004

[C98] Conklin, E., “Standardized application development for smart card technology”, Software Development Magazine, http://www.sdmagazine.com/documents/s=818/sdm9809d/, 1998

[F00] Frost, S., “The European Market of Smart Cards”, http://www.aboutit.de/view.php?ziel=/00/12/24.html, December, 2000

[SCA03] SmartCardAlliance, “Visa, Sony and Infineon Form Strategic Alliance for Smart Cards to develop a single-chip for Global Platform multi-function smart cards that support FeliCa and other industry standard contactless interfaces”, http://www.smartcardalliance.org/industry_news/industry_news_item.cfm?itemID=699

2.1.7 Sensors

With sensors, we mean the different elements that are capable of measuring a physical variable and transmitting it towards more intelligent devices through different networks. These sensor elements range from small electronic devices to quite sophisticated embedded systems.

Within the scope of a home, the application of a sensor may be included, among others, in one of the following groups:

- **Comfort variables***:* These relate to measuring temperature, humidity, CO_2, pressure and so on, i.e., ambient conditions. These measures can be done not only inside but also outside the home.
- **Technical security detectors***:* For security reasons, there are some variables that need to be checked in some special locations (e.g., water leakage in kitchens and bathrooms, gas leakage in the kitchen, fire detectors etc.).
- **Security and safety***:* This includes detectors of anti-intrusion, volumetric detectors, surveillance video systems, cameras, gate detectors and so on.
- **Welfare and health***:* There are various devices and sensors that improve people’s quality of life (e.g., a medical alarm medallion for elderly people, blood pressure meters, glucose meters etc.).
- **Voice capturing devices***:* One of the main attributions to ambient intelligence is the feature that communication between user and home devices is similar to human to human communication.
- **People detection***:* Different types of sensors exist to detect the presence of people inside the home. These sensors include voice sensors, face image capture devices, finger footprint sensors, eye iris detectors etc.
- **Diverse sensors***:* Not only inside but also outside the home there are other very diverse sensors that may help to give information about the environmental situation of the home (e.g., humidity detectors in the garden, light sensors on the windows, rain detectors in the roof etc.).

Most of the above sensor groups are needed to create an ambient intelligence home. For example, comfort variables are clearly useful to create applications to set ambient conditions according to the profiles of users. Applications like heating, air conditioning and air renovating are really appreciated by the user to create a comfortable atmosphere. Security and safety and technical security detectors are important to secure the home against water, gas, fire and also against intruders. Welfare and health sensors can take care of infants, elderly people, and handicapped or ill people. Therefore, it is possible to include applications and services around this subject.

There are some critical requirements on sensors within the context of ambient intelligence:

- Sensors should be spread all around the home (ubiquity is achieved thanks to that).
- All of them should interact with some intelligent device that processes the information that they capture. This communication should be wireless to avoid cabling, favouring mobility and configurability, and be close to a final product.
- Sensors must be hidden, in such way that their appearance is invisible to the user.
- Sensors should be as autonomous as possible regarding power consumption.

All the above challenges are achieved by means of commercial elements, having the main function of getting relevant information in real-time, mainly regarding the information about users (who they are; what they are doing; how they are). This is crucial for the success of ambient intelligence in general and Amigo in particular.

2.1.8 Actuators

An actuator is an element that has the possibility to interact with users, offering them some signals, information, physical magnitudes etc. This definition can be applied from the simplest version of actuators (simple on/off output to give power of a command signal to another device) up to the most complex ones (TVs, screens, climate devices etc.). For instance, if the microphone is considered a kind of sensor, the correspondent actuator would be a loudspeaker. if the sensors are the elements that perceive physical information, the user conditions, the orders from users, the input data and events, and if home intelligence is the element that processes all this information, the actuators are the elements that offer the result of this process to the environment. Furthermore, the home intelligence in an Amigo scenario is a mixture of the user and the system intelligence.

Within the scope of a home, applications for actuators include:

- Visualization actuators for applications related to entertainment, security, Internet access, information displays, home decoration etc. (e.g. TVs, different kinds of displays and wall panels, PDAs, WebPads, TablePCs, and Smart displays).
- Comfort actuators (e.g. air-conditioning, radiators).
- Housework support actuators (e.g. household appliances: washing machines, dishwashers, fridges, hobs, ovens and some parts of a household appliance are actuators as well: cold generator, water bombs, heat generator, motors).
- Lighting: each light generator is a final actuator, independent of its nature.
- Small and simple actuators (e.g. on/off outputs: power or signal).
- Others (e.g. blinds, windows).

2.2 Networks

The networked home integrates a number of networking technologies, in particular building upon wireless and wired (§2.2.1) networks over which communication is IP-based (§2.2.2) and possibly benefits from ad-hoc routing (§2.2.3) and real-time exchange (§2.2.4).

2.2.1 Home networks

The home network can be built up from different network segments. In a wired network, a device has to be connected to a cable infrastructure before this device can communicate with other devices in the network. Typical wired network technologies are: switched Ethernet, IEEE 1394, and HomePlug (using power lines).

Wireless communication media (IEEE 802.11, IEEE 802.15.x) have the advantage that no extra cables are needed. Within the wireless context, two types of devices can be considered: (1) portable devices (5 kilos or less), and (2) mobile devices. The second type of device (e.g., a PDA or Laptop) must be connected wirelessly, since this type of device is permanently on the move. In particular, mobile devices may be introduced into the home as guest devices with limited access rights. For the portable type, the wireless network is a convenience. Currently two wireless communication modes are known, infrastructure and infrastructure-less.

When wired and wireless connections exist together in the home, Access Points (APs) can connect the wireless devices to the wired infrastructure. One AP for an IEEE 802.11 based network may be sufficient depending on the network use and conditions. In the future, when home-network deployment increases, several interconnected APs are desirable or essential. The interconnection can be done in two ways: (1) cabling the APs, which brings us back to the cabling problem but in a limited sense, or (2) connecting them wirelessly which may be complicated by obstructing walls. In the future, more network connection points may be available in the home, e.g., introduced at construction time of the home, diminishing the disadvantage of a cumbersome plugging in to the wired home network. Currently, sharing the Internet connection available to a PC with other devices drives the set-up of an in-home network. Consequently, devices both wired and wireless connect to the Internet.

At the time of writing, the point of control of the in-home network has still not been determined. Many home networks employ devices that combine an Internet modem with wireless and wired connectivity and both firewall and routing functionality. When only one device combines wired and wireless connectivity it can be a central controller of the entire in-home network. For telecommunication and cable companies, this is an interesting model of operation. Another possibility is that a PC will take control of the entire home network. Yet, a third possibility is that the number of devices will be so large, of such a different nature or the network has such a topology that a central controller will not be accepted. In this case, the network has to be managed in a distributed way. Currently, all options are still open and it is necessary to investigate network management that can deal with those different situations.

Audio/video streams are important to the overall user experience in the home environment. Therefore it is an issue of concern to be able to provide the highest quality possible. Another issue is the quality as perceived by the user. Video coding experts use a value: Peak Signal to Noise Ratio (PSNR), to indicate the video quality of a rendered picture compared to the original reference picture. PSNR compares individual pictures and does not take the time aspect into account. Consequently PSNR is not always a good indicator for the quality of streamed video.

Table 2.1 shows some numbers [BF02] [C03] [CLB99] [HRBD03] [KLLN03] [K01] [V04] [BF02] [C03] [CLB99] [HRBD03] [KLLN03] [K01] [V04] to give an idea about the needed bandwidth values. Bandwidth on the communication media is mentioned versus the range or cable length (for all wired media, copper cabling is assumed).

Table 2.1 Measurements of communication medium bandwidth

Medium	**Range**	**Total bandwidth**	**Measured bandwidth**	**PER / Loss probability**[2]
Switched Ethernet	100 m	100 Mbit/s	90 Mbit/s 40 Mbit/s	0.02 0.0003
IEEE 1394	72 m	400 Mbit/s		< 10^-17
Homeplug 1.0	2 m 10 m 20 m	14 Mbit/s	4-6 Mbit/s 3-6 Mbit/s 3-6 Mbit/s	0.1
IEEE 802.11a	2 m 10 m 20 m	54 Mbit/s	18-24 Mbit/s 10-15 Mbit/s 6-7 Mbit/s	0.5
IEEE 802.11b	2 m 10 m 20 m 50 m	11 Mbit/s	5-6 Mbit/s 5-6 Mbit/s 5 Mbit/s n/a	0.5
IEEE 802.11g	2 m 10 m 50 m	54 Mbit/s	7-14 Mbit/s 6-8 Mbit/s n/a	0.5
Bluetooth	2 m	800 Kbit/s	570 Kbit/s	0.25
Ultra Wide Band	10 m	100 Mbit/s	n/a	n/a

From the Table 2.1, it can be concluded that switched Ethernet will easily support a number of video streams. Dependent on the path of the streams through the switches, losses may occur. Measures are needed to counter the effect of these losses on the video quality. Measures must also be taken to prevent bandwidth starvation. A 10 Mbit/s Ethernet cable can support 2 to 3 reasonable/medium quality SDTV video streams or 1 stream of high quality.

Homeplug, IEEE 802.11a and IEEE 802.11g performances are disappointing over distances of 10 meters and more. They come close to the low IEEE 802.11b performance. A wireless link will support at most one medium quality video over a short distance with current technology. The promises of UWB still need to be proven.

It is our belief that larger high quality screens will be connected to a video source through a wire. Small mobile screens and medium quality portable screens (17 inch) are more likely to be well served by a wireless connection.

Wireless network technologies are further examined in the next subsection.

2.2.1.1 Wireless networks

To enable wireless access in the Amigo Home environment, some different access technologies such as Wi-Fi, Bluetooth and UWB need to be examined.

Wi-Fi, also known as IEEE 802.11, comprises a set of wireless LAN standards developed by working group 11 of IEEE 802 to enable wireless communication among the members of a WLAN. The 802.11 protocol family currently includes three separate protocols that focus on data transmission (a, b, g). A low security level was originally included within these protocols (WEP encryption protocol), but is now part of another standard extension as 802.11i, which provides a higher security level to the network. Other standards in the family (c-f, h-j, n) are service enhancement and extensions, or corrections to previous specifications.

802.11b, with a maximum data rate of 11 Mbps, is the most widely accepted wireless networking standard, followed by 802.11a and 802.11g, which have data rates of 54 Mbps.

[2] Probability of packet loss is difficult to estimate given the many masquerading techniques at link layers.

The used frequencies are in the microwave range (2.4 GHz) for b and g and 5 GHz for a. The main feature of these standards is that they work in frequencies with minimal governmental regulation, and licenses to use this portion of the radio spectrum are not required in most locations.

There are two possible types of Wi-Fi networks: peer-to-peer (also called ad-hoc mode) and the so-called infrastructure mode:

- Peer-to-peer: this mode is a method for wireless devices to directly communicate with each other. Operating in peer-to-peer mode allows wireless devices within range of each other to discover and communicate in a peer-to-peer fashion without involving central access points.
- Infrastructure mode: this mode of wireless networking bridges a wireless network to a wired Ethernet network. Infrastructure mode also supports central connection points for WLAN clients. A wireless access point is required for infrastructure mode wireless networking, which serves as the central WLAN communication station.

Unlike many other wireless standards, the **Bluetooth** wireless specification includes both link layer and application layer definitions, which support data, voice, and content-centric applications. Radios that comply with the Bluetooth wireless specification operate in the unlicensed, 2.4 GHz radio spectrum ensuring communication compatibility worldwide. These radios use a spread spectrum, frequency hopping, and full-duplex signal at up to 1600 hops/sec. The signal hops among 79 frequencies at 1 MHz intervals to give a high degree of interference immunity. Up to seven simultaneous connections can be established and maintained. Bluetooth is intended to replace the cable(s) connecting portable and/or fixed electronic devices by using short-range radio links. It is envisaged that it will allow for the replacement of the many proprietary cables that connect one device to another with one universal radio link. Its key features are robustness, low complexity, low power and low cost. Designed to operate in noisy environments, the Bluetooth radio uses a fast acknowledgement and frequency-hopping scheme to make the link robust. The Bluetooth radio modules operate in the unlicensed ISM band at 2.4GHz, and avoids interference from other signals by hopping to a new frequency after transmitting or receiving a packet.

Ultra wideband (UWB) technology can be defined as any wireless transmission scheme occupying a bandwidth of more than 1.5 GHz. UWB has been around since the 1980s and has been used primarily for radar-based applications. However, recent developments in high-speed switching technology are making UWB technology more attractive for low-cost consumer communications applications. Industries, such as Intel and IBM, are currently working on internally funded research projects whose intent is to further explore the potential benefits and future challenges associated with extending UWB technology into the high-rate communications arena. UWB radio is a revolutionary communications mechanism that uses high-frequency microwave pulses for transmitting digital data over a wide spectrum of frequency bands with very low power intensity. Data can be transmitted at very high rates (for wireless local area network applications) and very low rates (for telemetry applications). Within the power limit allowed under the current FCC regulations, UWB radios can carry large amounts of data over a short distance, at very low power. In addition, there is the ability to carry signals through doors and other obstacles that tend to reflect signals at narrower bandwidths and at higher power levels. At higher power levels, UWB signals can travel significantly greater distances. Instead of transmitting traditional sine wave signals, UWB radios broadcast digital pulses timed very precisely on a signal across a very wide spectrum. The transmitter and receiver must be coordinated to send and receive pulses with an accuracy of trillionths of a second. Very high-resolution radars and precision (sub-centimetre) radio location systems can also use UWB technology.

UWB devices work inside the same increasingly crowded radio frequencies that many other systems use. They send out short electromagnetic pulses that last a half a billionth of a second, followed by pauses that are perhaps 200 times that length. By spreading the pulses over a wide area of the spectrum (roughly 1 GHz), the devices use extremely low power and little total bandwidth.

References:

[BF02] Buchan, S., Fulton, P., “Measurements of Bluetooth Performance in the Presence of Interfering Devices”, TN 4187, Philips, Redhill, 2002.

[C03] Caelers, R., “IP End-2-End Project, Network Test Results Throughput for UDP traffic”, Philips research note, draft, June 4 2003.

[CLB99] Cheng, S., Lai, K., Baker, M., “Analysis of HTTP/1.1 Performance on a Wireless Network”, Stanford University, Technical Report: CSL-TR-99-778, February 1999.

[HRBD03] Heusse, M., Rousseau, F., Berger-Sabbatel, G., Duda, A., "Performance Anomaly of 802.11b", IEEE Infocom, San Francisco, March-April, 2003.

[KLLN03] Katar, S., Lin, Y., Latchman, H.A., Newman, R.E., "A comparative performance Study of Wireless and Power Line Networks", IEEE communications, Vol. 41, Nr. 4, pp. 54 – 63, April, 2003.

[K01] Kazantzidis, M., "Wireless Adaptive Multimedia using Network Measurements", UCLA CS Technical Report 200102, February, 2001.

[WR03] WR Hambrecht + CO, "New wireless opportunities home networking focus", http://www.wrhambrecht.com/, March 2003.

[V04] Verhoeven, R., "Private communication", TU Eindhoven, 2004.

IEEE 802.11 Wireless LAN standard http://www.ieee802.org/11

Wi-Fi Alliance (IEEE 802.11 Trade Association) http://www.wi-fi.com

Specification of the Bluetooth System, Version 1.1 www.bluetooth.com

Intel home page htt.p://www.developer.intel.com/technology/ijt/q22001/articles/art_4a.h

IEEE1394 Technology http://www.askfor1394.com/

IP-based protocols

The Internet Protocol (IP) is a data-oriented protocol used by source and destination hosts for communicating data across a packet-switched network. The Internet Protocol provides an unreliable datagram service (also called best effort) that makes almost no guarantees about packets. They may arrive damaged, out of order, duplicated or may be dropped entirely (if the application needs reliability, this must be added by the transport layer).

Today's networks, and especially wireless networks, are quickly evolving toward broadband, offering higher bandwidth. At the same time, these networks are also moving toward all-IP networks. Internet Protocol Version 6 (IPv6) [GTB03] is particularly the mainspring of this movement since it is designed to run well on high performance networks (e.g., Gigabit Ethernet, OC-12, ATM) and at the same time still be efficient for low bandwidth networks (e.g., wireless). It indeed provides a platform for new Internet functionality that will be required in the near future, such as 128-bit source and destination addresses, improved addressing and routing capabilities and autoconfiguration.

Since many protocols are based on IP, we will focus on IP based protocols for ambient intelligence related to mobility, security, and configurability.

Two sorts of **mobility** can be considered in IP: macro- and micro-mobility.

- **Macro-mobility** involves a terminal (node) moving between two domains (geographically contiguous regions) that fall under the administration of completely distinct organizations. The two domains must collaborate to complete the handoff and to conduct authentication, authorization and accounting activities between the domains. The principal approach to support macro-mobility is Mobile IP [P02].
- **Micro-mobility** is the simplest form of mobility. A node is moving within a single domain, such as an enterprise or the communication range of a WLAN. Micro-mobility essentially involves intra-domain handoffs. There is no need for external coordination. Different protocols tackle this mobility, such as CIP [SGCW00] [CGKT00], HAWAII [RVST02], HMIP (MIP extension) [SCMB04], TeleMIP [CMDM01].

Security in the IP-based environment relies on two specific groups of mechanisms. The first group corresponds to authentication and the second one is devoted to securing data transmissions. These IP-based security environments are only part of a global security policy.

AAA stands for Authentication, Authorization and Accounting and focuses on network access requirements. AAA is a more general framework in the context of IP-based networks opposed to a specific protocol [LGGV00]. Many implementations of AAA IP-based protocols exist, such as Terminal Access Controller Access Control System (TACACS) [F93], Remote Authentication Dial in User Service (RADIUS) [AC03] and Diameter [CLGZ03] [L03]. IPsec [KA98] is a secure tunnelling technique that protects one or more paths between two hosts.

IPsec is designed to provide interoperable, high quality cryptography-based security for IPv4 and IPv6 (and transitively for MIP). IPsec is now considered to be robust. In the context of an ambient network, there are two major drawbacks that should be carefully studied. The first one is related to the set up of an IPsec tunnel which is not really flexible and far from auto-configuration compliant. The second one is the link to the overhead induced by the protocol, which may not be compatible with the bandwidth of some wireless links.

Configurability is a major issue in IP-based environments and, in the context of ambient intelligence a self-configuration scheme may be interesting. Some solutions particularly tackle automatic IP addressing, interoperability and administration. The Dynamic Host Configuration Protocol (DHCP) [D97] is devoted to the automated allocation, configuration and management of IP addresses and TCP/IP protocol stack parameters. This topic is no longer a research subject therefore DHCP solutions may involve some interoperability problems. Obtaining a pointer to a named server is also a critical configuration operation, and can be done through the Dynamic Name Service (DNS) protocol [M87] [DI04]. DNS goals include the preservation of the capabilities of the host table arrangements (especially unique, unambiguous host names) and the creation of a robust, hierarchical, distributed, name lookup system. Zero Configuration Networking [G01] improves network ease-of-use by making it possible to take laptop computers, connect them with a crossover Ethernet cable, and have them communicate using IP, without needing human administration. IPv6 provides an auto-configuration mechanism for terminal hosts. This mechanism does not apply to routers. While manual configuration should be acceptable for an edge router in the Internet, it is not acceptable in the context of ambient network where a mobile device can become a gateway between two IP networks, playing the role of a router. In this context, the output of the no longer active Zerouter working group might be interesting.

In ambient intelligence, many applications interact in a common environment. Multimedia applications are, for example, coming closer to the end user, in terms of proximity (e.g., Multimedia Message Service (MMS) in phones), and in terms of large-scale availability (e.g., Voice over Internet (VoI) or videoconferences with webcams). Each terminal supporting these applications must be identified (in the sense of naming). A great tendency is to attribute an IP address to all communicating terminals and to use specified standardized and optimized IP-based protocols, such as the Real-time Transport Protocol (RTP) [SCFJ03] for the real-time transmission of audio and video over unicast and multicast UDP/IP, Telephony Routing over IP (TRIP) [RSS02] for routing voice over IP through SIP proxies or H.323 gatekeepers, or the IP Multi-Media Subsystem (IMS) [3GPP04] for session control and applications services over wireline IP, 802.11, 802.15, CDMA, and GSM/EDGE/UMTS. Thus, IP and all associated protocols cannot be ignored in the ambient intelligence context, in general, and in Amigo in particular.

As many IP-based protocols are standardized, the main two challenges for Amigo are to:

- Identify the needs according to the use-case scenarios and choose the appropriate protocols. Developing new protocols will be too costly and contrary to a standardized approach.
- Ensure the interoperability of all these protocols. These interoperability and consistency problems are hard to solve if we only ake the level where these protocols are applied into account (most of the time, the network and transport level), as it implies the definition of new standards, a rigid evolution etc. A better approach is to ensure the interoperability at an upper level, by considering the IP-protocols as interchangeable building blocks. The definition of an open middleware using IP-based protocol building blocks should be considered.

References:

[3GPP04] 3GPP, “Service requirements for the Internet Protocol (IP) multimedia core network subsystem (IMS)”, Stage 1, TS 22.228, September 2004.

[AC03] Aboba, B., Calhoun, P., “RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)”, RFC 3579, 2003.

[CLGZ03] Calhoun, P.; Loughney, J.; Guttman, E.; Zorn, G.; Arkko, J., “Diameter Base Protocol, RFC 3588”, September 2003.

[CGKT00] Campbell, A. T., Gomez, J., Kim, S., Turanyi, Z., Wan, C.-Y., Valko, A., “Design, Implementation and Evaluation of Cellular IP”, IEEE Personal Communications, Special Issue on IP-based Mobile Telecommunications Networks, June/July, 2000.

[CMDM01] Chakraborty, K., Misra, A., Das, S., McAuley, A., Dutta, A., Das, S. K., ”Implementation and Performance Evaluation of TeleMIP”, In Proceedings of IEEE International Conference on Communications, Helsinki, Finland, Vol. 8, pp. 2488-2493, June, 2001.

[LGGV00] de Laat, C., Gross, G., Gommans, L., Vollbrecht J., Spence, D., “Generic AAA Architecture”, RFC 2903, August, 2000.

[D97] Droms, R., “Dynamic Host Configuration Protocol”, RFC 2131, March 1997.

[DI04] Durand, A., Ihren, J., “DNS IPv6 Transport Operational Guidelines”, BCP 0091, RFC 3901, September 2004.

[F93] Finseth, C., “An Access Control Protocol, Sometimes Called TACACS”, RFC 1492, July 1993.

[GTB03] Gilligan, R., Thomson, S., Bound, J., McCann, J., Stevens, W., “Basic Socket Interface Extensions for IPv6”, RFC 3493, February 2003.

[G01] Guttman, E., “Autoconfiguration for IP Networking: Enabling Local Communication”, IEEE Internet Computing, pp. 81-86, May/June, 2001.

[KA98] Kent, S., Atkinson R., “Security Architecture for the Internet Protocol”, RFC 2401, November 1998.

[L03] Loughney, J., “Diameter Command Codes for Third Generation Partnership Project (3GPP)”, Release 5, RFC 3589, September, 2003.

[M87] Mockapetris, P. V., “Domain names - concepts and facilities, Domain names - implementation and specification”, STD 0013, RFC 1034, 1035, November 1987.

[P02] Perkins, C. (Ed), “IP Mobility Support for IPv4, RCF 3344, August, 2002.

[RVST02] Ramjee, R., Varadhan, K., Salgarelli, L., Thuel, S., Wang, S. -Y., La Porta, T., “HAWAII: A domain-based approach for supporting mobility in wide-area wireless networks”, IEEE/ACM Transactions on Networking, vol. 10, no. 3, pp. 396 – 410, June, 2002.

[RSS02] Rosenberg, J., Salama, H., Squire, M., “Telephony Routing over IP (TRIP)”, RFC 3219, January 2002.

[RSCJ02] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E., “SIP: Session Initiation Protocol”, RFC 3261, June 2002.

[SCFJ03] Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., “RTP: A Transport Protocol for Real-Time Applications”, STD 0064, RFC 3550, July, 2003.

[SGCW00] Shelby, Z. D., Gatzounas, D., Campbell, A., Wan, C. -Y., “Cellular IPv6”, Internet Draft, November, 2000.

[SCMB04] Soliman, H., Catelluccia, C., El Malki, K., Bellier, L., “Hierarchical Mobile IPv6 mobility management (HMIPv6)”, Internet Draft, June, 2004.

2.2.3 Ad-hoc routing and hybrid protocols

Mobile Ad-hoc NETworks (MANET) [CM99] have primarily been used for tactical network related applications in the military context. A MANET is formed dynamically by an autonomous system of mobile nodes that are connected via wireless links without using the existing network infrastructure or centralized administration. The nodes are free to move randomly and organize themselves arbitrarily. Thus, the network’s wireless topology may change rapidly and unpredictably. It creates a suitable framework to address these issues by providing a multi-hop wireless network without pre-placed infrastructure and connectivity beyond Line of Sight (LOS).

In the context of MANET, the aim of the networking protocols is to use the one-hop transmission services provided by the enabling technologies to construct end-to-end (reliable) delivery services.

The sender needs to locate the receiver inside the network. Three services have to be provided by an ad-hoc protocol. First a mechanism to dynamically map the logical address of the (receiver) device to its current location in the network is needed. Then, routing and forwarding algorithms must be provided to route the information through the MANET. Finally, mobility of both the sender and receiver has to be handled by the protocol.

Numerous routing protocols and algorithms have been proposed, and their performance under various network environments and traffic conditions have been studied and compared. The cast property may be used to preliminary classify ad-hoc routing protocols: they use unicast, geocast, multicast, or broadcast forwarding:

- The most basic family is the **broadcast** mode where each message is transmitted on a wireless channel, received by all neighbours located within one-hop from the sender, and forwarded again. This naive flooding protocol may cause a broadcast storm problem due to redundant re-broadcast. Schemes have been proposed to alleviate this problem by reducing redundant broadcasting [SW04].
- **Unicast** forwarding means one-to-one communication, i.e., one source transmits data packets to a single destination. This is the largest class of routing protocols found in ad-hoc networks.
- **Multicast** routing protocols come into play when a node needs to send the same message, or stream of data, to multiple destinations.
- **Geocast** forwarding is a special case of multicast that is used to deliver data packets to a group of nodes situated inside a specified geographical area. Nodes may join or leave a multicast group as desired, and, on the other hand, nodes can join or leave a geocast group only by entering or leaving the corresponding geographical region.

MANET routing protocols are typically subdivided into two main categories: proactive routing protocols and reactive on-demand routing protocols:

- **Proactive** routing protocols are derived from legacy Internet distance-vector and link-state protocols. They attempt to maintain consistent and updated routing information for every pair of network nodes by propagating, proactively, route updates at fixed time intervals. As the routing information is usually maintained in tables, these protocols are sometimes referred to as table-driven protocols. Representative proactive protocols are the Destination-Sequenced Distance-Vector (DSDV) protocol [PB94], the Optimized Link State Routing (OLSR) protocol [JMQ98] and the Topology Dissemination Based on Reverse-Path Forwarding (TBRPF) protocol [BOT01].
- **Reactive** on demand routing protocols, on the other hand, establish the route to a destination only when there is a demand for it. The source node through the route discovery process usually initiates the requested route. Once a route has been established, it is maintained until either the destination becomes inaccessible (along every path from the source), or until the route is no longer used or expires. Representative reactive routing protocols include the Dynamic Source Routing (DSR) protocol [JM96], the Ad-hoc On Demand Distance Vector (AODV) protocol [PR99], the Temporally Ordered Routing Algorithm (TORA) protocol [PC97] and the Associativity Based Routing (ABR) protocol [DRWT97].

Using ad-hoc protocols in Ambient Intelligent environments mainly raises two challenges:

- The profusion and great variety of optimized protocols forces the identification of the right ones according to the needs expressed in use-cases scenarios.
- As no assumption about embedded ad-hoc protocols can be made (or only about a class of protocols), adaptation capacity is required to switch from one ad-hoc protocol to another one. Some strategic solutions can be found from **hybrid networks protocols**.

Routing in a hybrid network may follow different strategies. The first one consists of applying an ad-hoc routing protocol to the whole hybrid network, considering the infrastructure network as a static ad-hoc one and handling micro-mobility as ad-hoc mobility [BMA02]. If one of its strong points is simplicity, all mobility being handled by the ad-hoc routing protocol, it presents several drawbacks, especially regarding scalability and the need for appropriate fast mobility support. Some authors propose a second strategy where the hybrid network is, in term of routing, split in two entities, the infrastructure network and the ad-hoc one [WP00]. Routing in the ad-hoc network is handled by a classical ad-hoc routing protocol and the micro-mobility support is provided by Cellular IP (CIP), which manages routes to the ad-hoc mobile nodes in the infrastructure network. Since two different routing protocols are used, each one will use a channel multicast address in order to broadcast control packets without any interference between the two areas. There exists a third alternative strategy where the addressing architecture enables the partition of the hybrid network into several logical sub-networks in which different routing protocol instances may be applied [CF03].

Among the several sub-networks, one maps the infrastructure network and the remaining ones are each associated to one base station and its depending ad hoc nodes. Note that with this approach, we can deploy different types of routing protocols in each channel since all ad-hoc sub-networks will not necessarily have the same mobility pattern (dense zone, static or pseudo static backbone). An example is to use CIP in the infrastructure channel and different instances of proactive routing protocols in the different ad-hoc channels. Forwarding of the routing control packets is limited to one channel by using channel multicast addresses and not ad-hoc multicast ones. In consequence, a routing protocol operates in only one channel and diffusion of its routing information is limited to its channel.

The home automation environments considered in Amigo are composed of different networks such as the wired backbone (e.g. Ethernet), wireless networks (e.g. WLAN) or personal networks (e.g. Bluetooth) etc. This network list is not exhaustive and may probably change in the near future following technology evolution. Interaction of these networks constitutes hybrid networks. So, the first solutions in this field that were previously detailed should be carefully examined.

References:

[BOT01] Bellur, B., Ogier, R. G., Templin, F. L., “Topology broadcast based on reverse-path forwarding (TBRPF)”, Internet Draft, March 2001.

[CM99] Corson, S., Macker, J., “Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations”, RFC 2501, January 1999.

[DRWT97] Dube, R., Rais, C., Wang, K.-Y., Tripathi, S., “Signal stability based adaptive routing for ad hoc mobile networks”, IEEE Personal Communications, pp. 36-45, February, 1997.

[JMQ98] Jacquet, P., Muhlethaler, P., Qayyum, A., “Optimized Link State Routing Protocol“, Internet Draft, November 1998.

[JM96] Johnson, D. B., Maltz, D. A., “Dynamic source routing in ad hoc wireless networks”, Mobile Computing, pp. 153-181, 1996.

[PC97] Park, V. D., Corson, M. S., “A highly adaptive distributed routing algorithm for mobile wireless networks”, In Proceedings of INFOCOM, April, 1997.

[PB94] Perkins, C., Bhagwat, P., “Highly dynamic destination sequenced distance-vector routing (DSDV) for mobile computers”, Computer Communications Review, pp 234-244, October, 1994.

[PR99] Perkins, C., Royer, E. M., “Ad-hoc on-demand distance vector routing”, In Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, February, 1999.

[SW04] Stojmenovic, I., Wu J., “Broadcasting and activity scheduling in ad hoc networks, Mobile Ad Hoc Networking”, IEEE/Wiley, pp 205-229, 2004.

[BMA02] Benzaid, M., Minet, P., Al Agha, K., “Integrating fast mobility in the OLSR routing protocol”, In Proceedings of the IEEE International Conference on Mobile and Wireless Communication Networks (MWCN), Stockholm, Sweden, September, 2002.

[CF03] Chelius, G., Fleury, E., “Design of an hybrid routing architecture”, In Proceedings of the Fifth IEEE International Conference on Mobile and Wireless Communications Networks (MWCN), Singapore, October, 2003.

[WP00] Wijting, C., Prasad, R., “Evaluation of mobile ad-hoc network techniques in a cellular network”, In Proceedings of the IEEE Conference on Vehicular Technology (VTC), pages 1025–1029, 2000.

2.2.4 Real-time protocols[3]

Real-time protocols are communication protocols that support traffic with temporal guarantees. In the 80s, a lot of research has been conducted on these protocols in the industrial context. The objective was to introduce predictable behaviour in industrial local area networks to support multiperiodic, cyclic,

[3] Parts taken from Amigo SOTA, Jean-Dominique Decognitie, CSEM, updated by Philips Research.

and sporadic traffic with bounded latency. In the 90s, the advent of voice and video communication over the Internet has triggered a lot of research on the definition of new protocols that could support such traffic. The emphasis was more on the ways to guarantee throughput rather than mastering latency. This is a fundamental difference in the definition of “real-time” in communications. A second difference lies in the scale of the networks in which real-time communication takes place. Multimedia communication is meant to operate on the Internet whereas real-time industrial communications run locally most of the time on a single link, without routers or gateways.

Guaranteeing time constraints implies adequate protocols at every layer of the OSI model from the data link layer to the application layer. On the Internet, this is not yet achieved even if the IETF has defined a number of protocols such as SIP (Session Initiation Protocol), RTP (Real-Time Transport Protocol), RSVP (Resource Reservation Protocol), DiffServ, IntServ and others.

A network protocol developed for transporting video streams in real-time is RTP [SFCF96]. This is an application-level protocol, which is based on UDP (User Datagram Protocol) [P80]. An alternative transport protocol is TCP (Transmission Control Protocol) [P81]. RTP is preferred for real-time streaming over TCP. It uses timestamps to order packets and removes packets that are too late. In case of bandwidth limitations, the streaming will continue and the video frames are rendered at the specified times. TCP provides reliable delivery through acknowledgments and packet retransmission. For video streaming the consequence is that no packets are lost but the video can stall or retard. When the video source is a DVD player that can be paused, this behaviour can be acceptable. However, for live transmission or a DVD transmission that cannot stall, frames are lost due to buffer underflow. In that case TCP provokes both stalling video, accompanied by artefacts. For wireless networks, the number of losses may be as high as 30% leading to many stalls and artefacts when TCP is used.

Real-time protocols are considered necessary in the home networking environment not only for multimedia applications but also for other kinds of interactions that require bounded response times. This is a direct consequence of multimodal interactions. It may safely be assumed that a combination of both wired and wireless technologies will be used.

References:

[P80] Postel, J., “User Datagram Protocol”, RFC 768, Information Sciences Institute, 28 August 1980.

[P81] Postel, J., “Transmission Control Protocol”, RFC 793, Information Sciences Institute, September 1981.

[SFCF96] Schulzrinne, H., Fokus, G.M.D., Casner, S., Frederick, R., Jacobson, V., “RTP: A Transport Protocol for Real-Time Applications”, Internet Engineering Task Force, A/V Transport Working Group, January 1996.

2.3 Operation Systems

Various operating systems are eligible for the nodes of the networked home environment. However, we more specifically consider the major operating system architectures in use today, i.e., Linux (§2.3.1), Windows (§2.3.2) and Symbian (§2.3.3).

2.3.1 Linux

Linux is an Operating System with a special feature, it is licensed under the GNU General Public License, which guarantees that it is free, its source code is open, and every user can distribute it (opposed to copyrighted is copylefted). In practice, it is a collaborative project based on the volunteer work of thousands of software programmers, to whom Linux really belongs.

A distribution of Linux is a collection of software applications bundled with the core OS software (the kernel). Different distributions contain different bundled application and different versions (revisions) of the kernel. There are many Linux distributions, each with its unique flavour (Debian, Mandrake, Suse, Red Hat, Fedora etc.) and oriented towards some kind of users.

Linux is based on a philosophy that advocates the free flow of ideas in the creation of applications. One of the key motivating factors in application development under this scheme is that there is a chance to improve on what others have written. Teamwork builds a strong sense of community and this community manifests itself in many ways (newsgroups, Web sites etc.) with a shared knowledge base and free software for download.

The “Free Software Foundation” concerns itself with providing non-licensed alternatives to any license-restricted software. Its crowning achievement has been the GNU Project, which, among other things, provides core applications for Linux.

Open Source, on the other hand, describes a relatively recent initiative. It has gained significance with open source projects such as Mozilla, based on the liberated source code of the popular Netscape Web-browser, the broadly used Apache HTTP server and obviously, the Linux operating system and other related projects.

There are several advantages and disadvantages of using Linux. The advantages include:

- Cost: the Linux OS is entirely cost- and license-free.
- Stability: Linux almost never freezes under normal use.
- Support: there are several newsgroups devoted to answering your questions; extensive documentation is available.
- Portability: Initially developed for i386 based PCs, today, Linux also runs on ARM, Alpha, Sparc, 68k, MIPS, PowerPC, and other platforms.
- Power and customisation: Linux makes full use of a computer and can be tailored to specific hardware and software needs.

The disadvantages include:

- Interface: traditionally Linux has sported a command line interface; at least this is changing (e.g. KDE, Gnome).
- Support: there is no central body supporting Linux; but timely support can be bought through companies.
- Maintenance: Linux requires learning to perform administration tasks (rather true for any OS).
- Technical nature: Linux is not for everyone; it may require learning and experience to fully understand it depending on interest and motivation.
- Accountability: the use of Linux is strictly at the user’s own risk.

Due to the open oriented scope of the Amigo project, using an open operative system appears as an important issue within this project. Linux, due to its free and open source nature, seems to be an important option to be considered.

Several Linux distributions should be studied in the Amigo project to choose the most suitable ones, depending on the different devices and components to be used within the Amigo architecture. While some devices and components of the architecture find a standard distribution suitable such as Debian, Red Hat or Suse, other devices, such as embedded or mobile ones, would require more specific versions of the Linux Operating System.

References:

Linux Home Page http://www.linux.org

Relevantive’s Linux usability study http://www.relevantive.de/Linux-Usabilitystudy_e.html

Yankee Group TCO study http://www.yankeegroup.com/public/news_releases/LinEx Home Page http://www.linex.org/

Homepage of Slashdot http://slashdot.org

2.3.2 Windows

Windows is an operating system that powers the majority of personal computers today. It has a number of things built into the operating system such as a rich user interface, networking, security, media playing and recording capabilities, Internet browsers and a variety of other features. Windows is also extensible and supports adding new hardware and drivers, as well as adding new applications. There are a number of tools that facilitate the creation of new software applications for Windows.

Windows is packaged in a number of ways. It has specific versions for servers, business, home machines, media centres and pen computing. Although all of these have different functionality that leverages their specific usage scenario, they all contain a common code base, which makes applications easily run across them.

Advantages of Windows:

- Very common platform (defacto standard)
- Easily extensible
- Familiar user interface, very easy to use for people who aren't computer experts
- Official and community support
- Built in support for UPnP media servers and other media capabilities

Disadvantages:

- Has to be purchased per client
- No source code available

References:

The usability of everyday technology: emerging and fading opportunities http://portal.acm.org/citation.cfm?id=513667&coll=portal&dl=ACM&CFID=8745503&CFTOKEN=37406551

Windows Media Center Edition Information and Scenarios http://www.microsoft.com/windowsxp/mediacenter/evaluation/default.asp

Windows XP Home Edition Information http://www.microsoft.com/WindowsXP/home/default.asp

Microsoft Developers Network http://msdn.microsoft.com/

2.3.3 Symbian

Symbian is an embedded operating system that particularly targets smartphones. It is a market leader in this area, rivalled only by Microsoft's Phone Edition operating system. It comes pre-installed on the smartphone models of, e.g. Nokia and Sony-Ericsson. It supports Java development environments (i.e. J2ME runtime environment), and native Symbian applications developed in C or C++. Also scripting languages like Python have been ported to Symbian. The number of Symbian phones that are shipped increases by about 100% per year; in 2004 14.4 million handsets supporting Symbian were shipped (source: http://www.symbian.com/press-office/2005/pr050214b.html). Symbian has multiple sequential (upwards compatible) releases that are distinguished by their support of new J2ME services, media formats, network capabilities, and device capabilities.

The operating system contains support for all kinds of telecommunication services (including telephony, SMS, MMS), personal networking devices (e.g. Bluetooth, infrared, USB), and networking services like IP and HTTP over both the cellular networks that are supported by the phone (e.g. UMTS, GSM, GPRS) or over local network interfaces like wireless LAN. It also supports different categories of devices: touch screen phones (e.g. Sony Ericsson like), phones with miniature keyboards (e.g. Nokia communicators) or phones with only a numerical keyboard.

Advantages of Symbian for Amigo include:

- Common on a large number of smart phones, making it easy to demonstrate interoperability between the mobile domain and other domains
- Easily extensible and programmable
- Official and community support
- Supports various multimedia formats and DRM solutions, making it also an ideal personal device for viewing in-home content.

Disadvantages include:

- No source code available.
- It does not have native support for UPnP, although commercial products are available that support it.

References:

Symbian website http://www.symbian.com

3 Service oriented middleware

Middleware is a software layer that stands between the networked operating system and the application, providing well-known, reusable solutions to frequently encountered problems like heterogeneity, interoperability, security, and dependability. Most famous middleware infrastructures were introduced in the early 90s to facilitate the development of distributed systems and were based on the object-oriented paradigm (e.g., CORBA, Java-RMI, and DCOM). Using an object-based middleware infrastructure, distributed applications and higher-level middleware functionalities may be developed in terms of distributed objects using the middleware core infrastructure that offers basic functions for managing object lifecycle and remote interactions. Although this significantly eases the development of distributed applications, application developers still have to devise solutions for the enforcement of non-functional properties like dependability and persistence management. This has thus led to the introduction of distributed component technologies in the end of the 90s, which introduce the notion of a container. A container is an object host that enforces key non-functional properties for business applications like transaction and security management. Middleware technologies have further evolved towards service-oriented computing in the early 2000s so as to support the development of open distributed applications over the Internet. This allows packaging applications as autonomous services for access and possible composition with other applications over the Internet. In general, middleware-based software architectures constitute a significant enabler of the ambient intelligence vision. A number of research challenges that arose for the software systems realizing ambient intelligence applications can be tackled at the middleware-level.

This chapter concentrates on technological development in the area of middleware architectures for the home environment, addressing base service oriented architectures for the home (§3.1), discovery protocols (§3.2) and their interoperability (§3.3), which are key to the automatic and dynamic configuration of the open networked home environment that must in particular deal with security and privacy (§3.4), QoS management (§3.5), content management (§3.6), and accounting and billing (§3.7).

3.1 Service oriented architectures

Building open distributed systems that allow interaction among evolving heterogeneous devices has always been a complex task. Developers face several challenges such as interoperability, resource management, synchronization, performance issues, providing security, scalability, dependability etc. Specifically, an open system composes autonomously implemented and administrated software systems, which communicate over a public or private network. The composed systems may be implemented in different programming languages and deployed on different software and hardware platforms.

Enabling the composition of autonomous software systems requires defining interaction protocols agreed upon by the composed systems, i.e., the communication protocols to be followed and their behaviour need to be understood and adhered to by all the composed parties although the protocols implemented by the resulting composite system cannot be fixed at design time. Various software technologies and development models have been proposed over the last 30 years for easing the development and deployment of distributed systems (e.g., middleware for distributed objects). However, the generalization of the Internet and the diversification of connected devices have led to the definition of a new computing paradigm: the **Service-Oriented Architecture (SOA),** which allows developing software as a service delivered and consumed on demand. The benefit of this approach lies in the looser coupling of the software components making up an application, hence the increased ability for making systems evolve as, e.g., application-level requirements change or the networked environment changes.

The SOA approach appears to be a convenient architectural style towards meeting one of the key objectives of the Amigo project, that is, to enable interoperability among heterogeneous applications deployed on the heterogeneous devices of the networked home environment. This section provides an overview of SOA principles together with that of the most popular existing software technologies complying with the SOA architectural style, i.e., OSGi (§3.1.1), UPnP (§3.1.2), and the Web services architecture (§3.1.3). From this study, it appears that current software technologies do not meet the interoperability requirements for the Amigo system.

Drawbacks include:

- Little support of a specific core middleware platform to ensure interoperability at the communication level,
- Little support for semantic-based interoperability, hence dealing with interaction between services based on syntactic description for which common understanding is hardly achievable in an open environment,
- Centralized solution that does not scale in general and affects openness.

The second issue may be addressed using semantic modelling, as undertaken in the context of the Semantic Web, while the latter issue may be tackled based on appropriate service composition.

3.1.1 OSGi

The OSGi Alliance is an open standardization organization formed by Sun Microsystems, International Business Machines, Ericsson and others in March 1999 (after it was first called the Connected Alliance). Over the past few years, it has specified a Java programming language-based service platform that can be remotely managed. The core part of the specifications is a framework that defines an application life cycle model and a service registry. Based on this framework (see Figure 3.1), a large number of OSGi services have been defined: Log, Configuration Management, Preferences, Http Service (runs servlets), XML Parsing, Device Access, Package Admin, Permission Admin, Start Level, User Admin, IO Connector, Wire Admin, Jini, and UPnP. Specifications are developed by the members in an open process and made available to the public free of charge and without licensing conditions. The OSGi Alliance has a compliance program that is open to members only. Currently at least 12 implementations compliant with one of the three successive specifications R1, R2, R3 are available.

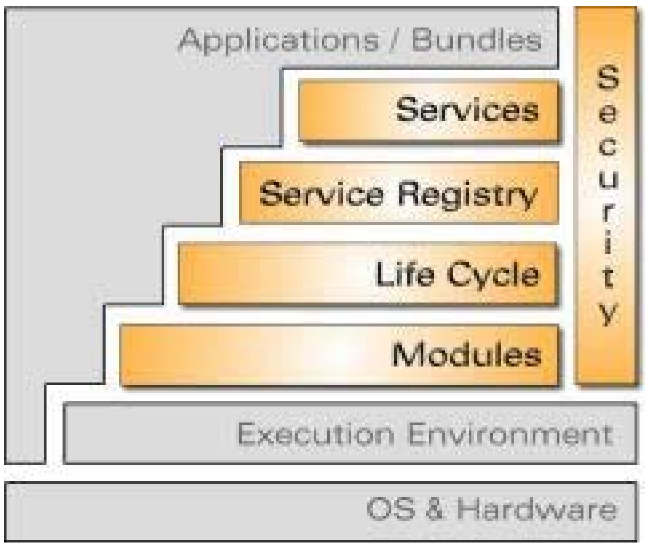

Figure 3.1 The OSGi framework

The original focus was on Service Gateways but the applicability turned out to be much wider. In 2003, the well known Eclipse IDE selected OSGi as the underlying runtime for their plug-in architecture. The Equinox project experimented with this idea, and built the runtime for Eclipse R3 that has been available since December 2003. In October 2003, Nokia, Motorola and other OSGi members formed a Mobile Expert Group (MEG) that will specify a service platform for the next generation of smart mobile phones, addressing some of the needs that MIDP cannot manage. The application areas of the OSGi Service Platform are currently as various as service gateways, cars, mobile telephony, industrial automation, building automation, PDAs, grid computing, white goods, entertainment, fleet management and IDEs.

OSGi Framework

The OSGi Framework implements a dynamic component model. Applications (called bundles) can be remotely installed, started, stopped, updated and uninstalled without restarting the framework. Life cycle management is defined in APIs, which allow the remote downloading of management policies. The service registry allows bundles to detect new services, or leave the services and adapt accordingly.

OSGi Bundles

A bundle is a Java JAR archive including Java class files and any other necessary data the service might need, including possibly native code. A special manifest file provides information about resources needed or provided by the bundle. Some bundles contain an 'activator' which allows the bundle to be 'started' and 'stopped' as an application. A bundle without an activator must be considered as a library that provides (exports) a list of packages. OSGi specifies a strict class loading policy so that classes provided by a bundle cannot be used in another bundle unless specified in the import-export clauses of both bundles. Life cycle management (see Figure 3.2) is one of the most prominent features of the OSGi framework. It provides the necessary mechanisms to allow remote management in a wide variety of management models, and is also accessible via an API that allows bundles to manage other bundles (e.g. install, start).

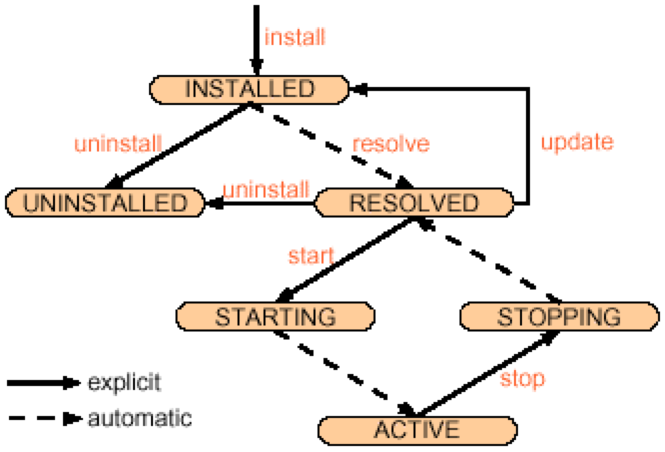

Figure 3.2 The OSGi bundle life cycle

Service Registry

The service registry allows bundles to co-operate by providing the basic functions:

- Register objects with the Service Registry
- Search the Service Registry for matching objects
- Receive notifications when services become registered or unregistered

Objects registered with the Service Registry are called services. Services are always registered with an interface name and a set of properties. The interface name represents the intended usage of the service. The properties describe the service to its audience. For example, the OSGi Log Service would be registered with the org.osgi.service.log.LogService interface name and could have properties such as vendor=acme. Discovering services is done with notifications, or by actively searching for services with specific properties. A simple, but powerful, filter language is used to select exactly the services that are needed.

The OSGi framework supports LDAP query syntax, with operators such as *and* (&), *or*(|), *not* (!), *less* (<=), *greater* (>=), *approximate* (~=), *equals* (=), *substring* (*), *presence* (*) etc.

Security

One of the goals of the OSGi Service Platform is to run applications from a variety of sources under strict control of a management system. A comprehensive security model, present in all parts of the system, is therefore a necessity. The OSGi specifications use the following mechanisms:

- Java 2 Code Security
- Minimized bundle content exposure
- Managed communication links between bundles

Java 2 Code Security provides the concept of permissions that protect resources from specific actions. For example, files can be protected with file permission classes. Permission classes take the name of the resource and a number of actions as parameters.

OSGi Implementations

Oscar is an open source (GPL) implementation of the OSGi framework specification, currently compliant with a large portion of the R3 specifications. It proposes several interesting features like the 'service binder' and the 'Oscar Bundle Repository'.

The Knopflerfish project is based on the Gatespace GDSP OSGi framework, which has been in development since 1999. Knopflerfish is absolutely R3 compliant. Moreover, it comes with a visual desktop, which allows the management of the whole framework. Knopflerfish is available under a BSD style license.

Physalis is an OSGi implementation for the .NET (Compact) Framework. The starting date of the activity for this project is August 2004, so the status is yet pre-alpha and there is no documentation or downloads available. However, the fact of being the first open source (LGPL) OSGi implementation for the .NET framework is reason enough for making this brief comment about Physalis in this review.

JEFFREE stands for Java Embedded Framework FREE. The last version (0.91 from 3 March 2003) provides almost complete OSGi v2.0 specification. An interesting feature is that Jeffree is compatible with personal Java, whereas Oscar and Knopflerfisch require Java 2. This allows Jeffree to be installed on a large variety of devices. However, the project development seems to be stopped for the moment. Jeffree is available under the LGPL license.

Commercial OSGi Implementations

Ubiserv (Gatespace Telematics) is a complete certified OSGi release 3 compliant service platform. Ubiserv can be viewed as the commercial Knopflerfish distribution. This product replaces Gatespace Telematics' former OSGi products GDSP and SGADK.

The IBM Service Management Framework, (IBM SMF) an implementation of the OSGi Service Platform specification, is designed to execute properly under all OSGi-defined execution environments and conforms to OSGi Service Platform Release 3.

The Avelink embedded gateway (atinav) is R3 compliant, with full support for personal Java.

The ProSyst mBedded Server, R3 compliant, is available in several editions (infotainment, smart home, mobile devices) and has been tested against a variety of embedded device operating systems.

The JVMA complete list of certified implementations is available on the OSGi website.

Interoperability with other devices

The UPnP Device Architecture specification (see §3.1.2) and the OSGi Service Platform provide complementary functionalities. The UPnP Device Architecture specification is a data communication protocol that does not specify where and how programs execute. That choice is made by the implementations. In contrast, the OSGi Service Platform specifies a (managed) execution point and does not define what protocols or media are supported. The UPnP specification and the OSGi specifications are fully complementary and do not overlap. The same applies to Jini and OSGi. OSGi Release R3 defines standard ways of incorporating UPnP and Jini technologies on OSGi platforms.

The Jini Driver module in the OSGi framework is responsible for the Jini-to-OSGi and OSGi-to-Jini transformations. Using this API, OSGi services from the framework can be exported easily to the Jini network (see §3.2.2), and Jini services from the Jini network can be imported into the OSGi framework.

Concerning UPnP, the OSGi specification framework therefore defines how an OSGi bundle can implement a service that is exported to the network via the UPnP protocols, and how to find and control UPnP devices that are available on the local network.

Interest for Amigo

OSGi offers a simple component model that makes it a good candidate for the deployment of Amigo services. It already specifies a number of framework APIs (e.g. service registry, event service) and services (http, log, UPnP etc.) while keeping open the possibility to either provide new implementations for these services or define new services. It is fully complementary to distributed architecture specifications like UPnP or Jini.

OSGi is available not only on residential gateways, but also on PCs and embedded Java-enabled devices. Therefore, delivering Amigo applications and libraries as OSGi bundles could help to the construction of an open and modular platform.

References:

Avelink, definition of OSGi http://www.avelink.com/OSGI/

IBM Service Management Framework http://www-306.ibm.com/software/wireless/smf/

Jeffree homepage http://jeffree.objectWeb.org

Knoplerfish project http://www.knopflerfish.org

The object Web consortium http://www.objectWeb.org

Oscar- OSGi framework implementation http://oscar.objectWeb.org

OSGi alliance Web site http://www.osgi.org

Physalis project home page https://developer.berlios.de/projects/physalis

Homepage of ProSyst www.prosyst.com

Gatespace Telematics, Ubiserv http://www.gatespacetelematics.com/userarea/login/ubiserv.shtml

3.1.2 UPnP

Universal Plug and Play (UPnP) is a technology and industry consortium driven by the UPnP Forum, including more than 700 companies from consumer electronics, computing, home automation, home appliances and related industries. The UPnP Forum defines UPnP Device and Service Descriptions, which are based on open, Internet-based communication standards for interoperability, according to a common Device Architecture contributed by Microsoft. Similar to the Internet, UPnP is based on wire protocols, not APIs, hence it is independent of the underlying OS, programming language and physical medium.

UPnP enables devices to join automatically to a network, find and use networked devices and services provided by one another without manual configuration or involvement in a peer-to-peer fashion. It describes and supports the discovery and communication between devices to find and use services; specifically focusing on but not restricted to PCs, Internet gateways, consumer electronics and PC peripheral devices.

UPnP has been recently gaining acceptance from consumer electronic and PC peripheral makers, and is becoming the common way for these devices to interact with more powerful computing devices (PCs, servers or handhelds). There are also specific industry efforts such as Digital Living Network Alliance (DLNA) that use UPnP as a base and ensure that specific device interaction scenarios are reliable enough for the home market.

Components of the UPnP Architecture

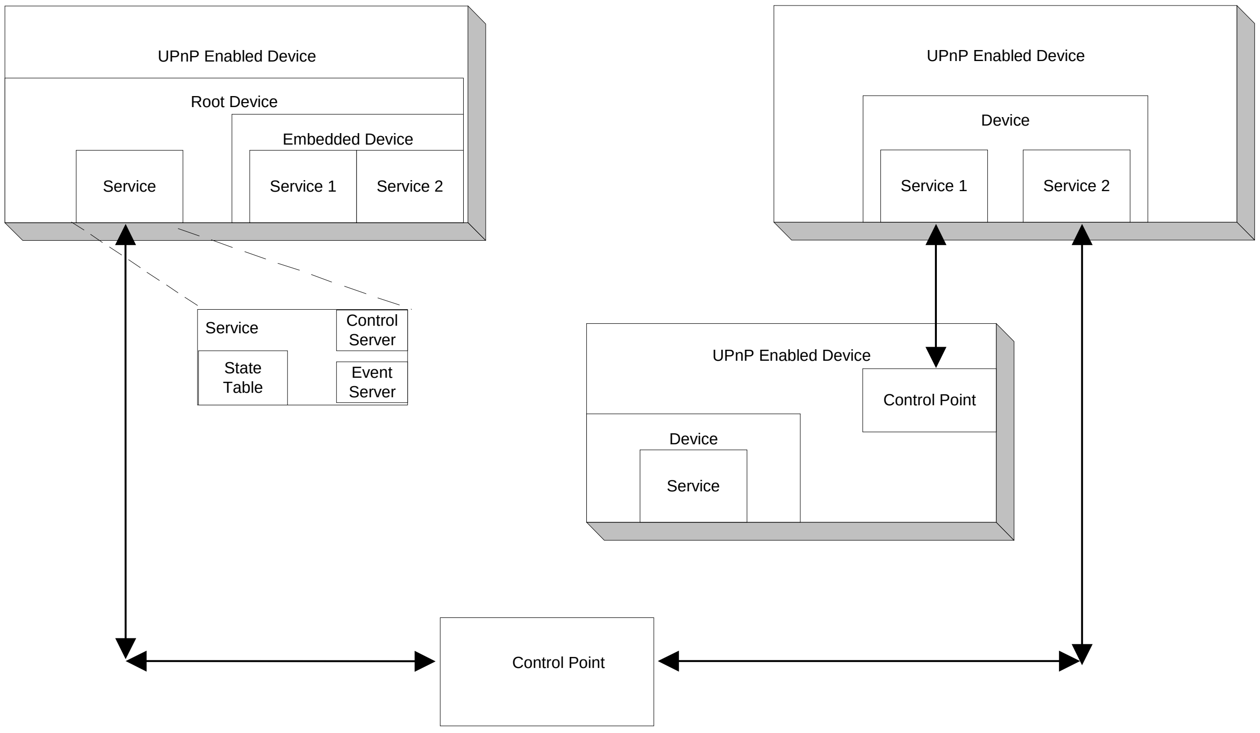

Figure 3.3 UPnP Control Points, Devices, and Services

The UPnP architecture is composed of three elements: Control Points, Devices and Services (see Figure 3.3).

Devices handle the discovery and control requests from Control Points and produce events to notify Control Points. A device can host several services and other embedded devices. For instance, a TV/VCR combo device would host a TV and VCR embedded device, which in turn would consist of a display service, a tape service, a tuner service, and a clock service. Services provided by a device type are standardized by different working groups and all information (device manufacturer, specific device type and version information as well as services provided by the device) about the device is stored in an XML device description file. There are a number of device types defined in the UPnP Forum covering things from printers and scanners, to media devices, network access points and lighting controls.

Services are the smallest units in the UPnP network, which expose actions and model state through state variables. These actions and state variables are stored in an XML service description file. The state of a service is modelled in the state table through state variables. Action requests of the Control Points are handled by the Control Server, and status changes are forwarded to interested Control Points through the Event Server.

A Control Point is used to discover and control the devices providing services on the network. After discovering a device on the network, a Control Point can:

- Retrieve the device description and get a list of associated services
- Retrieve service descriptions for interested services
- Invoke actions to control the service
- Subscribe to the service’s event source. Anytime the state of the service changes, the event server will send an event to the control point.

Protocols

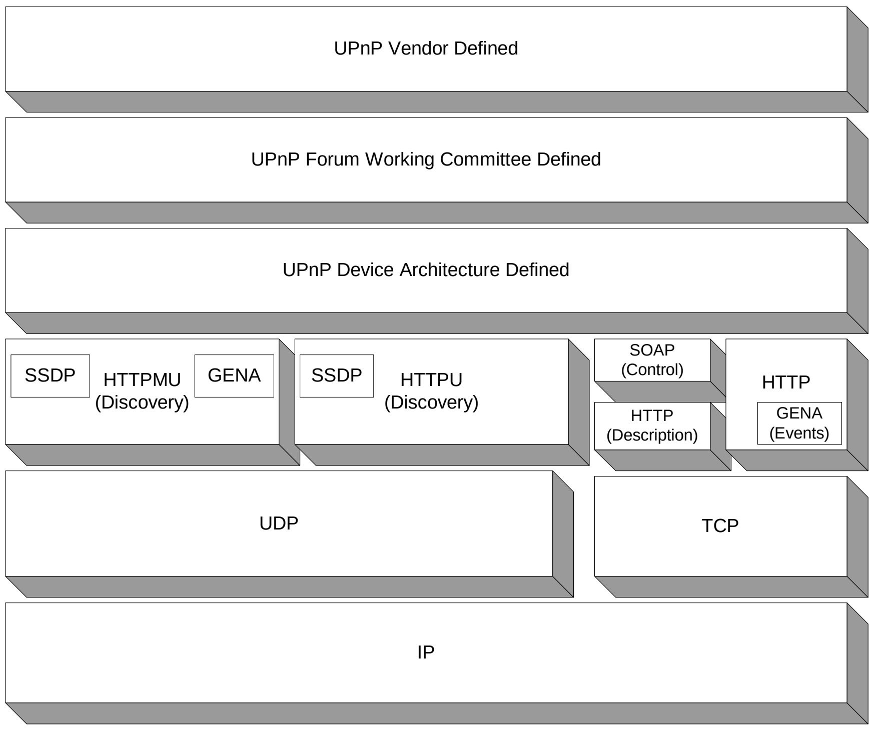

Figure 3.4 The UPnP Protocol Stack

UPnP uses the following protocols (see Figure 3.4):

- TCP/IP makes UPnP independent of the underlying physical media and guarantees interoperability between vendors
- HTTP is a core part of UPnP. HTTPU and HTTPMU are variants of HTTP, which support unicast and multicast messaging respectively on top of UDP/IP. These protocols are used in SSDP for service advertisement and discovery
- SSDP (Simple Service Discovery Protocol) defines how network services are discovered by control points and how devices announce themselves on the network (by using HTTPU and HTTPMU as the underlying communication mechanism)
- GENA (Generic Event Notification Architecture) is defined to provide the ability to send and receive notifications using HTTP over TCP/IP and multicast UDP
- SOAP (Simple Object Access Protocol) defines the use of Extensible Markup Language (XML) and HTTP to execute remote procedure calls. Each UPnP control request is a SOAP message that contains the action to invoke along with a set of parameters
- XML is a core part of UPnP used in device and service descriptions, control messages and events.

Networking Steps

Addressing – When a device is connected to the network, it searches for a DHCP host to acquire an IP address. If no DHCP server is available, intelligent Auto IP mechanisms are used to acquire an IP address.

Discovery – When a device is added to the network it advertises itself, and when a control point is added to the network, it can discover devices of interest by using SSDP.

Description – After discovering the device of interest, to get more information about the device, a control point can retrieve the device description. The UPnP Device description, written in XML, contains information required for the control, events and presentation steps.

Control – After getting the device description, to control the device, the control point can learn more about a device's services by retrieving the service description. To use a service, a control point sends a control message, and receives the runtime state of the service.

Events – When the state of a service changes, the control points that are interested in the service status are notified to avoid control inconsistencies.

Presentation – A device can also provide a presentation page through the description URL specified in the device description file, and depending on the page capabilities control points can use this page to control and view the status of the service.

References

UPnP Forum Web Site http://www.upnp.org/

3.1.3 Web Services

Today, with virtually every computer and potentially every application connected to a single network, there are increased opportunities for establishing relationships dynamically. To achieve this, transactions must happen in real time and also use open Internet standards to integrate these disparate islands of applications. Major IT players, vendors, service providers, and consumers of IT services have now come together to agree on a common standard - Web Services. Web Services based on open standards act as a facade to provide a uniform and widely accessible dynamic interface to expose operations. Figure 3.5 illustrates how a typical Web Service works.

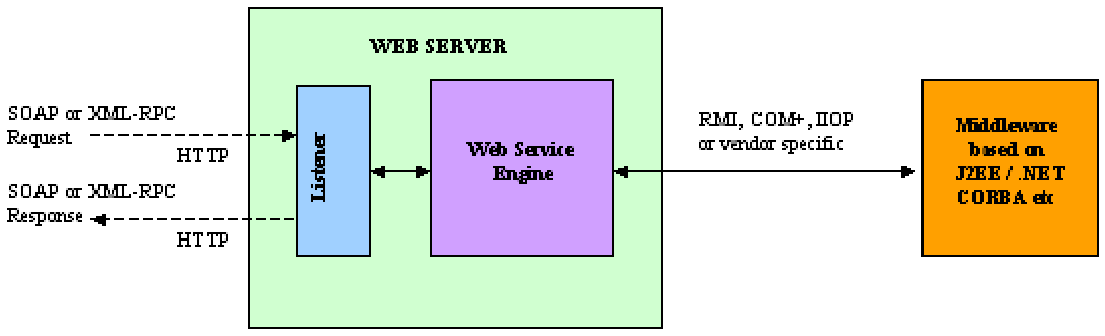

Figure 3.5 Workings of a typical Web Service

XML and HTTP form the core of Web Service standards. Web service technology stacks, composed of SOAP, WSDL, and UDDI are the basic building blocks that provide a solid infrastructure, enabling systems to interoperate with relative ease and reduced cost of integration. Refer to the References Section for more information on these technologies.

Web Services, based on a Service-Oriented Architecture (SOA), typically involve three entities: the service provider, the service broker, and the service requestor. Figure 3.6 shows the interaction of components that make up the SOA.

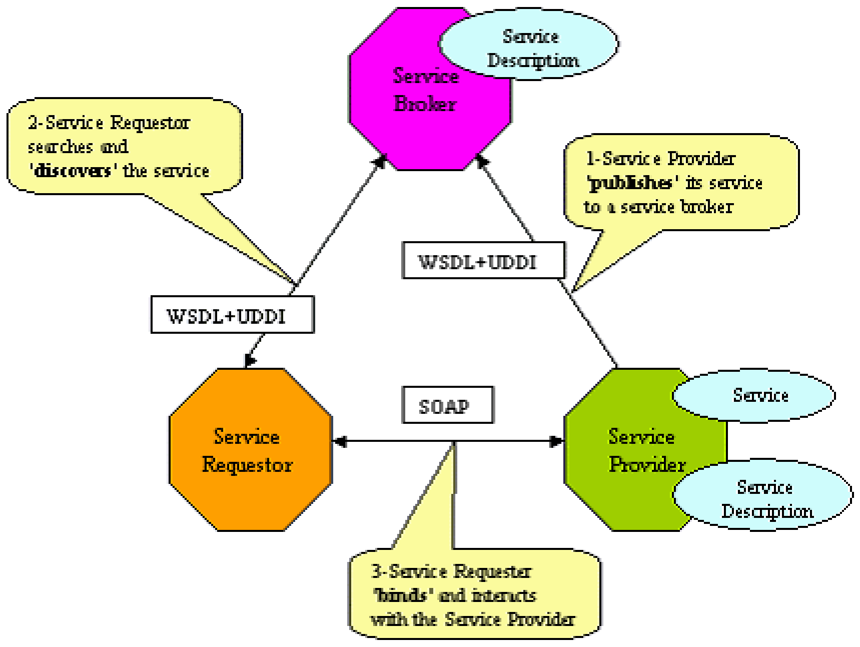
Service Broker
Service Description
2-Service Requestor searches and 'discovers' the service
1-Service Provider 'publishes' its service to a service broker
WSDL+UDDI
WSDL+UDDI
SOAP
Service Requestor
Service Provider
Service
Service Description
3-Service Requester 'binds' and interacts with the Service Provider

Figure 3.6 The Service-oriented architecture

In a typical Web Services scenario, a service provider publishes its services in a standard format on a global registry (the service broker) that is accessible over the Web. A service requestor can be an application on a company's intranet, a partner application on the Internet, or an application on an end-user device. It searches the registry to discover the service and its associated endpoints. It then binds with the service provider and invokes a method on the Web Service using the SOAP protocol. The method invocation is sent as an XML document embedded in an HTTP request. The service provider's Web server receives the HTTP request and passes it to the Web Service engine, which in turn invokes an appropriate method on the backend system. The result of the method invocation (formatted as an XML document) is embedded within an HTTP response and returned to the service requestor. The actual object implementing program logic on the backend system may be a Java class, an EJB, a .NET component, or any legacy application.

3.1.3.1 UDDI

UDDI (Universal Description, Discovery and Integration) is one of the core elements enabling Web services and is concerned with the publishing and discovery of Web services in the Internet. The UDDI specification defines both the registry API for publishing service information and for performing focused and unfocused discovery, and the specific format to use for some of the stored description information.

There have been three versions of the UDDI specification. The first version of UDDI was focused on the concept of a Universal Business Registry (UBR) that represents a master directory of publicly available e-commerce services (similar to a telephone directory). Public UBRs are maintained by a few third parties and are accessible to any entity that wants to advertise a service. The UDDI specification was updated in v2 to better take into account relationships between business entities. The major changes include better modelling support of the business relationships within a company (parent-child, peer-2-peer, and identity), better replication support, and an improved categorization of businesses and services by allowing externally checked taxonomies. New features have been added to UDDI v3 to fully support different types of UDDI registries (e.g., open or private registries managed by trade groups, companies, or communities). Specifically, UDDI v3 enables the association of roles to registries such as *root* and affiliate, which allows the modelling of partnerships between registries. UDDI v3 also enables the generation and management of distributed registration keys, and the use of XML digital signatures to ensure data integrity and ownership/entry validity. Finally, UDDI v3 added subscription so that changes in registries can be tracked, synchronously (API call) or asynchronously (callback to a SOAP endpoint).

The UDDI specification defines a set of data structures for describing businesses and their services. The major ones are the Business Entity that holds basic business information, the Business Service that represents a single service, the Publisher Assertion that establishes a relationship between two Business Entities, the Binding Template that contains pointers to technical descriptions of services as well as their access points, and the Technical Model, which is an abstract description of a particular standard, specification, or behavior to which the Web Service adheres. As WSDL has become the de-facto standard for describing Web Services, it has been integrated into the UDDI specification.

Web Service composition

The modularity and interoperability of the Web Services architecture enable complex distributed systems to be easily built by assembling several component services into one composite service. Composing Web Services relates to dealing with the assembly of loosely coupled autonomous components so as to deliver a new service out of the components' primitive services, given their corresponding published interfaces. The integration of Web Services is then realized according to the specification of the overall process composing the Web Services. Building new systems out of existing systems has the advantage of reusing existing applications. Moreover, different Web Services realizing the same functionalities can be integrated during runtime, based on their availability in a given environment, increasing the availability of the composite application. The composite system, which can also be exposed as a Web Service with an interface and appropriate deployment, can deliver new functionalities with emergent properties.

There are several approaches for dealing with the composition of Web Services, addressing different needs. Web Services may be composed within a closed environment. For example, an enterprise may have several Web Services for different tasks that may be composed for creating a more complex business process. The composition may also be realized in an open environment, where Web Services that are autonomous in functionality and in administration may be composed by a third party to achieve a certain task. The former type of composition may be realized following a top-down approach where the global composite scheme is well defined, and composed Web Services can then be built or existing ones integrated according to this definition. The latter one promotes a bottom-up development process where each existing Web Service describes its composition capabilities that enable them to be integrated later in different types of composite applications.

Several composition languages for specifying composite Web Services have been proposed. They cover different aspects of the composition such as the specification of the composition process from the perspective of the composite service (referred as to **orchestration**) and the specification of specific interaction protocols for multi-party interactions among several Web Services (referred as to **choreography**). Choreography describes message exchange rules for two or more interacting partners. Choreography aims at describing a specific task such as a business process, independently of any particular Web Service. Existing languages for specifying choreographies include the Web Service Choreography Description Language (WS-CDL) [W3C04], BPML [BPMI02] and the model presented in [BFHS03]. Orchestration descriptions give the complete process flow describing interactions with composed services, including the application logic of the composite service. The composition process can be specified as a graph (or process schema) over the set of composed Web Services. The specification of a composition graph is in general given with XML-based declarative languages such as BPEL [OASIS05], which directly support reuse, openness, and the evolution of Web Services by clearly distinguishing the specification of component Web Services (comprising primitive components that are considered as black-box components and/or inner composite components) from the specification of composition.

3.1.3.3 Semantic modelling

Semantics, the science of describing meaning in some machine-readable form, has been a popular research topic for decades. By making use of typical features of today’s World-Wide Web infrastructure, which was not designed to carry meaning at all, we might be able to create a knowledge platform which offers much more potential to change the information world than all previous attempts.

The current World-Wide Web is a syntactic web where the structure of the content is presented while the content itself is only readable by humans. Although the WWW has resulted in a revolution in information exchange among computer applications it still cannot fulfil the interoperation among various applications without some pre-existing, human-created agreements.

The next generation of the Web aims to alleviate such problem. The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

Ontologies are considered to be a key technology to make the Semantic Web become a reality [F00] [ALDS04] [CG00]. They play a pivotal role by providing a source of shared and precisely defined terms that can be understood and processed by machines. A typical ontology consists of a hierarchical description of important concepts and their relations in a domain, task or service. The degree of formality employed in capturing these descriptions can vary, ranging from natural language to logical formalisms, but increased formality and regularity clearly facilitates machine understanding. Therefore a powerful ontology language, which can help to formalize the Web is the most sought after thing in the Semantic Web. In a very simple way, ontologies can be seen as specifications or conceptualizations. In this usage, an ontology is a set of concepts, such as things, events, and relations, that are specified in some way (such as specific natural language) in order to create an agreed-upon vocabulary for exchanging information.

The most desired features for the ontology languages are that such a language should be well designed for the intuition of human users without loosing adequate expressive power. It should be well defined with clearly specified syntax and formal semantics and it should be compatible with existing Web standards. Various existing ontology languages, starting from RDF, and following the line with DAML+OIL and OWL are covered in the following.

The RDF (Resource Description Framework) is being developed by the W3C for the creation of metadata describing Web resources. A strong relationship stands between RDF and XML. In fact, they are defined as complementary: one of the goals of RDF is to make it possible to specify semantics for data based on XML in a standardized, interoperable manner. The goal of RDF is to define a mechanism for describing resources that makes no assumptions about a particular application domain nor the structure of a document containing information.

The data model of RDF (which is based in semantic networks) consists of three types:

- Resources (subjects), entities that can be referred to by an address on the WWW;
- Properties (predicates), which define specific aspects, characteristics, attributes or relations used to describe a resource
- Statements (objects), which assign a value for a property in a specific source.

Probably the easiest way of understanding these terms is “Resources have Properties with Values". These three elements make up the base element of the RDF model, the triple.

RDF Schema (RDFS) is a declarative language used for the definition of RDF schemas. The RDFS data model (which is based on frames) provides mechanisms for defining the relationships between properties (attributes) and resources. Core classes are class, resource and property; hierarchies and type constraints can be defined (core properties are type, subclassOf, subPropertyOf, seeAlso and isDefinedBy). Some core constraints are also defined.

In response to RDF shortcomings the DARPA Agent Markup Language (DAML) sprang from a U.S. government sponsored effort in August 2000, which released DAML-ONT, a simple language for expressing more sophisticated RDF class definitions than permitted by RDFS. The DAML group soon pooled efforts with the Ontology Inference Layer (OIL), another effort providing more sophisticated classification, using constructs from frame-based AI. The result of these efforts is DAML+OIL, a language for expressing far more sophisticated classifications and properties of resources than RDFS. DAML+OIL is a semantic markup language for Web resources. It builds on earlier W3C standards such as RDF and RDF Schema, and extends these languages with richer modelling primitives. DAML+OIL provides modelling primitives commonly found in frame-based languages. The language has clean and well-defined semantics. A DAML+OIL ontology is made up of several components, some of which are optional, and some of which may be repeated. A DAML+OIL ontology consists of zero or more headers, followed by zero or more class elements, property elements, and instances.

The OWL (Ontology Web Language) is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology.

OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDFS, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web. OWL is a revision of the DAML+OIL Web ontology language incorporating lessons learned from the design and application of DAML+OIL.

OWL provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users, *OWL Lite, OWL DL* and *OWL Full*. Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded.

References:

[BFHS03] Bultan, T., Fu, X., Hull, R., Su J., "Conversation Specification: A New Approach to Design and Analysis of E-Service Composition", Proceedings of the 12th International World Wide Web Conference, 2003

[BPMI02] Business Process Modeling Language, BPMI.org, http://www.bpmi.org, 2002

[OASIS05] OASIS, Web Services Business Process Execution Language, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel, 2005

[W3C04] Web Services Choreography Description Language Version 1.0, W3C Working Draft, http://www.w3.org/TR/ws-cdl-10/, 2004

[F00] Fensel, D., "Relating Ontology Languages and Web Standards, Informatik und Wirtschaftsinformatik", Modellierung, Foelbach Verlag, 2000.

[ALDS04] Arroyo, S., Lara B., Ying Ding, Stollberg, M., Fensel, D., "Semantic Web Languages – Strengths and Weakness", IADIS International Conference AC, Lisbon, Portugal, 2004

[CG00] Corcho, C., Gómez-Pérez, A., A., "Roadmap to Ontology Specification Languages", in proceedings of the 12th European Workshop on Knowledge Acquisition, Modelling and Management, Springer, Juan Les Pins, France, 2000

Description of the DAML+OIL ontology markup language http://www.daml.org/2001/03/reference

Introduction to DAML http://www.xml.com/pub/a/2002/01/30/daml1.html

OWL Web Ontology Language Overview http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1

Web Services on World Wide Web Consortium http://www.w3.org/2002/ws/

SOAP on World Wide Web Consortium http://www.w3.org/2000/xp/Group/

UDDI on Oasis http://www.uddi.org/

WSDL on World Wide Web Consortium http://www.w3.org/TR/wsdl

3.2 Service discovery protocols

One of the main subjects the IT industry is currently working on is the integration of automatic configuration into present and future systems. This trend is partly due to the requirements of mobility and the effortless use of consumer equipment. Mobility in particular means getting away from configured environments, into foreign networks with unknown services. However, a mobile device/user cannot predict such environments or take advantage of available services. Service discovery protocols enable users to perform tasks easily and quickly, e.g., plug printers, storage devices, speakers, PCs, intelligent appliances, wireless devices etc. directly into a network. As a consequence, every computer, device, and user on the network will know that a new device has been added and is available. Service Location Protocol (SLP), Jini and SSDP used by UPnP (for a complete overview of UPnP see 3.1.2) are three of the leading technologies in this area [HeA02] [DFKM02].

3.2.1 SLP

The Service Location Protocol (SLP) is an IETF standard for decentralized, lightweight, and extensible service discovery. It uses service URLs, which define the service type and address for a particular service. Based on a URL, users (or applications) can browse available services in their domain and select and use the one they prefer. There are three agents in SLP: the user, service and directory. The User Agent (UA) is a software entity that sends service discovery requests on a user application's behalf. The Directory Agent (DA) caches advertisements from Service Agents (SAs) and processes discovery queries from UAs. A SA advertises itself by registering with a DA. The SA periodically renews its registration with the DA, which caches the registration and sends an acknowledge message to the SA. A UA sends a service request message to the DA to request a service location. The DA responds with a service reply message that includes the URLs of all services matched against the UA request. Thus, the UA can access one of the services pointed to by the returned URL. In SLP, the DA is optional. A DA might not exist in a small network, in which case the UA service request messages are directly sent to the SAs.

3.2.2 Jini

Jini can be viewed as the next step after the Java programming language, towards making a network look like one large computer. Jini [HeA02] allows manufacturers to release devices that can attach to a network. Each pluggable device will define itself immediately to a network device registry. When it attempts to use or access a resource, it will be able to download the necessary code from it so as to communicate with it. No longer, will special device support software, known as a device driver, need to be present in an operating system. The operating system will know about all accessible devices through the network registry. Jini uses the Java Remote Method Invocation (RMI) protocol to move code around the network and supports remote events and transactions that help programmers write distributed programs in a reliable and scalable fashion. The heart of Jini is a triple of protocols: discovery, join and lookup. The first two–discovery and join are used when a Jini device enters the network. During discovery the device must locate a lookup service by multicasting a request on the local network or a remote lookup service known to it a priori. Once the lookup service is located, the device registers its service object and service attributes with the lookup service. This service object contains the Java programming language interface for the service supported by the device, including the methods that users and applications will invoke to execute the service, along with any other descriptive attributes. When a user or application wishes to use a service, a client in the user device requests the service by Java type and perhaps other service attributes. The lookup server ships a copy of the service object over the network to the client. Then, the client uses it to talk to the service. The client interacts directly with the service via the service object.

3.2.3 SSDP

The Simple Service Discovery Protocol (SSDP) is used by UPnP for service discovery. This protocol announces a device's presence to others and discovers other devices or services. In other words it is analogous to the trio of protocols in Jini. SSDP uses HTTP over multicast and unicast UDP. A joining device sends out an advertisement multicast message to advertise its services to control points. Control points function similar to Jini's lookup services. A control point, if present, can record the advertisement but other devices might see this multicast message directly. In contrast to Jini, SSDP can work without control points. SSDP makes extensive use of XML to describe device features and capabilities. The aforementioned advertisement message contains a URL that points to an XML file in the network that describes the device's capability. By retrieving this XML file, other devices can inspect the advertised device's features and decide whether it is important or relevant to them. A description for a service includes a list of actions to which the service responds and a list of variables that model the service state at runtime. The service publishes updates when these variables change, and a control point can subscribe to receive this information. Updates are published by means of event messages containing the names and values of one or more state variables.

3.2.4 Summary:

From the standpoint of enabling ambient intelligence in the networked home environment:

- Jini is a protocol based on Java technology, hence it shares all the advantages of it, namely platform independence, advanced security, ease of source code development and plenty of networking features. Embedded devices incorporating some type of Java Virtual Machine to meet the necessary prerequisites so as to exploit Jini. The integration of JVMs to mobile devices, especially to mobile phones and PDAs, is already a fact
- SSDP/UPnP offers high compatibility with Microsoft products, so we can assume support for the development process. Special features of UPnP are highly appreciated in light of the objectives of Amigo, such as, automatic IP assignment and configuration, functional capability in the absence of control points, which is particularly useful in spontaneous networks and the extensive use of XML for service description and discrimination.
- The Service Location Protocol, like UPnP, provides decentralized capabilities, e.g., the Directory Agent is not necessary. The implementation of the needed software modules is not restricted to a specific programming language, so the most suitable one for the particular case can be selected. It also offers the functionality of browsing the available services and searching based on keywords and logical operators. Other notable characteristics of SLP are the extensibility of protocol elements and operation over IPv6.

In general, service discovery protocols have respective advantages and drawbacks, and it cannot be considered that a single protocol will emerge for devices of the networked home environment. Instead, the networked home environment will integrate networked nodes that are heterogeneous in terms of hosted software infrastructure, including middleware and related functionalities such as the embedded service discovery protocol. Then, a major challenge for the Amigo system and in particular for the Amigo middleware is to enable interoperation among such heterogeneous devices, enabling inter-working of the various distributed middleware functionalities.

References:

[DFKM02] Dobrev, P., Famolari,D., Kurzke, C., Miller, B.A., “Device and Service Discovery in Home Networks with OSGi”, IEEE Communications Magazine, August, 2002

[G99] Guttman, E., “Service Location Protocol: Automatic Discovery of IP Network Services”, IEEE Internet Computing, July – August, 1999

[HeA02] Helal, S., “Standards for Service Discovery and Delivery”, IEEE Pervasive Computing, July-September 2002.

Interoperability

Among the previously presented systems, UPnP, Jini and Web Services provide distributed service-oriented architectures, whereas OSGi describes the service Oriented Architecture of a single platform, relying on separate standards to tackle the distribution issues.

UPnP, Jini and Web Services are not interoperable. Jini is reserved to Java programming environments, whereas UPnP and Web Services are programming language independent. Web Services and UPnP may both use SOAP as a protocol for remote invocation, but use different formats and concepts for service descriptions.

OSGi 3.0 Release R3 defines standard ways of incorporating UPnP and Jini technologies on OSGi platforms. Also, the Axis project provides tools for mapping between Java and Web Services. All these tools allow a client running in a OSGi platform to interact with UPnP devices, Jini services or Web Services, or a server running on an OSGi platform to make its services available to UPnP control points, to Jini clients or as Web Services.

However, the fact that a client can access an UPnP device, a Jini service or a Web Service by using standardized Java APIs does not solve the interoperability question. Some work has been done for UPnP/Jini Interoperability. However, as stated by the authors of [ACGR03], several issues remain.

The most important difference is service standardization. Each service type (printer, remote file service, LCD projection service etc.) will have a standardized Java interface and a standardized UPnP/XML specification, which may or may not offer equivalent functionality. Furthermore, standard types in Java, such as vectors, hash tables, etc. tend to find their way into Java interfaces, while UPnP service specifications tend to be based on simpler types. Moreover, UPnP working groups tend to define standard devices (grouping several services) whereas Jini working groups focus on services.

One of the goals of Amigo is to allow the interoperability of services and client applications based on different protocols. Amigo devices should be able to interact with heterogeneous devices and heterogeneous networks in the domains of domestic appliances and consumer electronics. Amigo devices should in particular be interoperable with consumer electronic devices following DLNA recommendations, which mean UPnP devices and control points.

Interoperability between UPnP, Jini and Web Services is still an open issue: if a client uses the Jini paradigm and looks only for Jini services, and there is a UPnP device providing such a service, the client will probably not be able to use the UPnP service, for two reasons. The first is that the client will not discover the service, as there is no automatic way to map service descriptions. The second is, once the service is found, how to interact with it. Full interoperability would require semantic matching between UPnP and Jini descriptions.

Amigo middleware will ensure the interoperability of "Amigo devices" with at least UPnP devices and control points. Concerning domestic appliances, communicating devices often use specialized non-IP networking. Special work will have to be done to integrate these devices into the Amigo network.

References:

[ACGR03] Allard, J., Chinta, V., Gundala, S., Richard III, G.G., “Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability”, Proceedings of the International Symposium on Applications and the Internet, 2003

Open Services Gateway Initiative http://www.osgi.org

3.4 Security and Privacy

In open distributed systems it is not possible to trust hosts on a network. For applications involving the transmission of privileged information, e.g. credit card numbers, techniques have been developed to protect the information. Techniques based on cryptography are the main tools used for all security issues since they enable the achievement, to some point, of most of the main principles of a secure system: confidentiality, integrity, accessibility, authenticity and demonstrability of origin. **Authentication** techniques, which are often based on cryptography, allows clients and servers to securely determine each other's identities, thus enabling access control and providing authenticity to the exchanged information. *Encryption* techniques provide confidentiality, integrity and demonstrability of origin. Digital Rights Management (DRM) is a growing concept, which includes techniques and measures for copyright protection, ownership assertion and digital content authentication.

Authentication is the process of reliably verifying the identity of someone or something. One authentication system is Kerberos, originally developed at MIT (US) but based on the work of Needham (Cambridge, UK) and Schroeder (US) [SNS01]. Kerberos is an industry standard mechanism and consists of a Key Distribution Centre (KDC) that runs on a physically secure node somewhere in the network, and a library of subroutines that are used by distributed applications that want to authenticate their users. Kerberos was designed so that once the user logs into the workstation by providing a name and password, then the workstation is able to obtain information from the KDC for any process to access remote resources on behalf of the user. Kerberos also generates temporary private keys, called session keys, which are given to two clients, allowing encryption of messages between those clients. Some projects have tried to improve on Kerberos, including the EU-funded project SESAME (Secure European System for Applications in a Multi-vendor Environment). SESAME uses similar techniques to Kerberos, and improves on it by providing better support for heterogeneous domains and security policies, more sophisticated access control features, better manageability, audit and delegation. The designers recognized that Kerberos was important and designed SESAME so that a world standard was possible incorporating features of both technologies. SESAME is a construction kit, composed of a set of security infrastructure components for product developers. It has been used in a number of industry products. Both of these authentication systems have proved to be successful, where authentication for distributed systems is needed.

Specific authentication techniques involve something that one is (Biometrics) something that one possesses (i.e. a card) or something that one knows (password). There exists plenty of commercial biometric authentication software and devices. Usually a stronger biometric authentication implies a more uncomfortable test for the user. The digital signature enables a strong, secure and comfortable authentication method based on challenge-response. The authentication system sends a non-repeating number to the object (i.e. a user, a device) and this signs the challenge providing also its certificate. Smart Cards can be used with these kinds of authentication methods. On the other hand, authentication based on passwords is usually considered a light method due to the often low password quality: a password quality help application can solve this problem. Kerberos and SESAME can be adapted to any of these authentication techniques.

Access control (often called **authorization)** determines which devices or services may or may not be used by a given user. A possible access control implementation is the one based on the Access Control Lists (ACL) used by for example the Windows operating system to grant or deny access to securable objects (i.e. services, files, devices and resources). This vision requires one or more configurable access control list per securable object, describing which actions may or may not be performed by a certain entity on that object. This solution provides a wide range of configuration possibilities.

Role Based Access Control (RBAC) is another access control approach. It was introduced in 1992 by Ferraiolo and Kuhn, and has gained prestige and popularity due to the complexity and cost reduction it provides in security administration for large networks. In a RBAC model access rights are grouped by role name, and access to resources is restricted to entities, who have been authorized to assume or have been directly associated to that role.

Encryption is the translation of data into a secret code and involves keys to encrypt information to be transmitted over a network and to decrypt the information for the intended recipient. Encryption may be symmetric or asymmetric. The main disadvantage of symmetric encryption techniques (AES, TripleDES, IDEA) is that they require shared key exchange through a secure channel but are light in terms of computation load. On the other hand, asymmetric encryption techniques (also known as public key encryption) provide confidentiality, integrity and demonstrability of origin without any secure key exchange, but are inefficient for ciphering larges amounts of data. Therefore, they are usually used for key exchange in a confidentiality-enforcing scenario. RSA's (US company) public key cryptography is a technology used in over 100 million copies of software from various commercial vendors. One example is SSL (Secure Sockets Layer), developed by Netscape (US company), which uses, among others, RSA encryption and includes the use of digital certificates. TLS is the IETF standardisation of SSL and provides almost the same functionalities of SSLv3.0. TLS (as well as SSL) is located on top of the transport layer and provides an interface for secure TCP socket connections to applications. Once a TLS/SSL session is established between a client and a server, different connections can be created without new parameter negotiations, preventing an excessive overload. Furthermore, to avoid computational overload, compression is also implemented by TLS/SSL.

WS-Security defines extensions for SOAP message headers to provide integrity, confidentiality and authentication at the message level. It enables security token association to messages, binary data or plain text. Although WS-security is not a security solution by itself and must be used jointly with other security protocols, it does provide a method to combine Web Services with a given security solution.

Digital Rights Management technologies deal with the problem of managing rights on digital content. There are several technologies that provided a solution to the illegal reproduction of copyrighted material to some extent: biometric authentication, time stamping, Smart Card technology, watermarking and others. Digital Transmission Content Protection (DTCP) is a standard for protecting content while being copied from a device to another. Based on cryptographic techniques, it assures that a protected content can be copied to a certain device only if it is authorised for receiving such content. The authorisation is determined on copy protection rules embedded on the media itself. The DTLA (Digital Transmission License Administrator) is responsible for assigning device certificates for registered devices.

References:

[DBF01] Deswarte, Y., Blain, L. and Fabre, J.-C., “Intrusion Tolerance in Distributed Systems”, in Symp. on Research in Security and Privacy, Oakland, CA, USA, 1991.

[FDR01] Fabre, J.-C., Deswarte, Y. and Randell, B. (1994). Designing Secure and Reliable Applications using FRS: an Object-Oriented Approach, in 1st European Dependable Computing Conference (EDCC-1), Berlin, Germany LNCS 852, 2001.

[FK01] Ferraiolo, D.F. and Kuhn, D.R., "Role Based Access Control" 15th National Computer Security Conference, 2001.

[SNS01] Steiner, J.G., Neuman, B. C. and Schiller, J. I., Kerberos: An authentication services for open network systems, In Proceedings of the Winter 1988 Usenix Conference, February 1988.

Web Services Security (WS-Security) Specification http://www-106.ibm.com/developerworks/webservices/library/ws-secure/

DTLA Home Page http://www.dtcp.com/

Netscape Corporation: The SSL Protocol http://wp.netscape.com/eng/security/SSL_2.html

Sesame group: Sesame homepage http://www.esat.kuleuven.ac.be/cosic/sesame3.html

RSA Homepage http://www.rsa.com/

3.5 QoS management

QoS management is a necessity for every environment. It concerns the ability to obtain service-level guarantees such as timeliness, availability, fault-tolerance, and survivability for applications executing in large-scale environments. The problem of meeting the QoS requirements of applications is made harder in a ubiquitous computing environment where new services are expected to be added into existing applications. In open distributed environments, the presence of services can thus quickly cause problems because of the interactions between services and resource sharing.

Current research on resource management and QoS aims to guarantee that each application will have enough resources to be executed with the required quality of service. However, existing mechanisms often rely on complex architectures or management models. Those include QoS specification mechanisms and the automatic discovery of application needs. The aim of these operations is to enable the environment to determine the needs of the application at run-time in terms of resources. Thus, these needs can be compared with the available resources to estimate if the required QoS can be achieved. Furthermore, according to the domain requirements, the QoS may be defined statically but also dynamically. It can be done either with negotiation mechanisms or with a system/application agreement. The advantage of a dynamic approach is the possibility to renegotiate the contract, or agreement, concerning the variations of the resource availability and adapt the environment behaviour.

Management mechanisms are very important for a distributed environment, since they allow the consistent management of resource reservation in a coherent way according to the provided service. Nevertheless, it requires supervision of the quality perceived by the user and the quality offered by the system. In doing so, it is possible to detect overuses in the resource consumption and react in consequence. With such mechanisms, allowing controlling component access to resources, the deployment environment can prevent illegal access to a resource, or excessive use. Today, lots of mechanisms exist. In any case, from a high-level point of view, existing approaches can be mainly split into three main models of resource control: annotations, parameterization of the execution environment, and declarations.

In the strict sense, an **annotation** is an "explanation or review coming along a text". In our context, an annotation is a way to add some useful information inside an application to verify and control its behaviour. Even if annotations can be added automatically (e.g. by a compiler), they are usually introduced by the programmer, the administrator or the user of the application. Consequently, rather than going through the implementation of an application, it is better to understand and observe its execution in order to identify provided services and resource consumption. Indeed, this type of analysis (e.g. used resources, how and where they are consumed) is critical for collecting interesting data and extracting pertinent information.

The aim of annotations is to realize an analysis that enables such information to be provided [PPQ02]. Another use for annotations is to provide information regarding optimization to a compiler. At invocation-time and run-time of the application, dedicated supervision functions are invoked when the control flow reached an annotation point.

Contrary to the previous approach, resource control can be realized within the execution environment rather than at the application level. Thus, the original application is never modified and no additional information is directly added. The **parameterization** principle consists in providing both a program and parameters to the execution environment. Concerning these parameters, the environment will be adapted and configured to come up with the QoS expectations. These parameters usually deal with restrictions and constraints on resource manipulation (e.g. access rights, quotas, use of threshold). They will only be considered at run-time. Nevertheless, this mechanism does not allow verifications to be added at admission-control time. Indeed, the information provided by the parameters is only relevant at run-time and thus only allows the configuration of the execution environment. Hence, it does not authorize the deployment of mechanisms of resource management at invocation-time of the application.

Another solution to the resource control problem is to enable the application to declare its expectations. These expectations can be in terms of resources, quotas, restrictions, or limitations. This **declarative approach** can be instantiated as rules defining restrictions from the component or the execution environment [BPO98]. However, in most cases, this behaviour remains limited since it does not allow adaptation of the QoS. Another solution is based on the notion of contract. Indeed, the application expressing its needs in terms of resources, asks for a specific service from the deployment environment. On its side, the environment, by accepting to host such an application, commits to provide the expected resources. A contract is “a convention by which one or more entities commit to give, to do, or not to do something toward one or more entities”. This principle is based on the specification of the quality in terms of an acceptable level of QoS wanted by each entity involved and is promised to be respected. Once needs have been specified by the application, the system begins the negotiation process. As a result, a contract between the application and its environment is established. If the contract is accepted, the system must guarantee that it has enough resources to execute the application with the expected level of QoS [S03].

One of the challenges for Amigo is to enforce QoS in terms of at least security and performance regarding resource consumption, i.e., the two mandatory criteria for the consumer acceptance of ambient intelligence systems. So it becomes critical to be interested in resource management handled by ambient computing services. Nevertheless, this cannot be done in a standard way and requires a high-level approach: at the applicatgion level.

References:

[BPO98] Barnes, J.F., Pandey, R., Olsson, R., “Supporting Quality of Service in HTTP Servers”, In Proceedings of the 17th Symposium on Principles of Distributed Computing (SIGACT-SIGOPS). Puerto Vallarta, Mexico, 1998

[S03] Le Sommer, N., “Contractualisation des ressources pour les composants logiciels: une approche réflexive“, University of Bretagne Sud, France, 2003.

[PPQ02] Pang, R., Peterson, L., Qie, X., “Defensive Programming: Using an Annotation Toolkit to Build DoS Software”, In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), Massachusetts, USA, 2002.

3.6 Content management

Content management is a term that is mainly used to describe the complete workflow that information undergoes (starting from content creation to the content delivery to the end-user) in a business context. It is developed around the different (business) stakeholders:

- Content creators or owners. These are the owners of the content. They either create the content on their own (e.g. film or TV studios) or own the rights to the content (e.g. national distributors)

- Content editors. Marketing organizations often modify or re-arrange content for branding or commercial reasons (e.g. commercial Web portals or TV stations)
- Content distributors. These are the owners of the usually large backbones that distribute or deliver the content towards the end-user (e.g. Internet providers, telecom operators or cable companies).

Each stakeholder applies specific technologies at each step in the process to get the desired result. Without going into the detailed technologies at each step, the overall goal of traditional content management can be described as a process to handle the creation, manipulation, storage and distribution of information in large networks.

Although traditional content management might not seem directly applicable to Amigo (besides the fact that all the business stakeholders would be interested in using it to bring content into the automated home), the automated home needs to solve the same issues albeit in a different and smaller environment.

Content that is brought into the home (either by own creation, renting or buying) needs to be stored and distributed towards different devices in the home. The different aspects of traditional content management (creation, manipulation, storage and distribution) are all applicable in a home scenario. The difference between traditional content management and content management for Amigo is that the latter should not be driven by business stakeholders and requirements, but by the user and the envisioned scenarios in an automated home. This implies:

- Content creation: content might be brought into the home on a physical medium, from the Internet, a device or from any other source
- Content storage: content needs to be stored so it can be retrieved at any time. Privacy and Security issues play an important role in this (traditional content management can rely on a secure zone without the need for different levels of access as in a home scenario)
- Content manipulation: content maybe needs to be manipulated to fit or suit the device it is being distributed to (change in format or size)
- Content distribution: content needs to be delivered to the device that is required to render it. There might be several paths to the device or resource constrains on the link towards that device.

The challenge for Amigo in relation to content management is to design and implement a process workflow specific for the home scenario. Limited or no research has been done up to today in this area since it was usually driven by business requirements. For Amigo, the user scenarios and derived requirements need to be carefully analyzed in the light of usability, performance, privacy, security and QoS.

3.7 Accounting and billing

The accounting and billing schemes [AAH00] [FDL00] used in the Internet have been quite simple until now. Users have been mainly billed with a flat rate, based on their subscription and/or the duration of their connection for accessing the Internet. In mobile telecommunication networks, users have been mainly billed based on their subscription and the call duration, as well as a number of other parameters (e.g., type of communication, location of the destination, and so on). The charging function collects information related to a chargeable event from several network nodes. The billing information generated by network nodes is structured in the form of a charging data record (CDR, inheritance from the old Call Detail Records) [BB00] and transferred via standard accounting protocols. In the near future these schemes are expected to receive extensive modifications as a consequence of recent technological advances combined with the emerging dominance of the Internet Protocol. The proliferation of devices supporting a variety of different services will generate unique billing requirements for providers as they leverage the information of subscribers to offer targeted services via connected devices. Given the anticipated demand for multimedia services, flat rates for unrestricted access to multimedia content will not be cost-effective. Flat rate pricing models may prove successful for individual services where the bandwidth and content cost per service is predictable, but for services where the cost is unpredictable, a usage-based pricing scheme must be leveraged to be profitable.

Accounting and billing have inspired a diverse set of researchers who are working toward the design of new frameworks and protocols. The Authentication, Authorization and Accounting (AAA) Working Group, created by the IETF, has focused on the development of requirements for authentication, authorization and accounting as applied to network access. The Authentication, Authorization and Accounting ARCHitecture Research Group (AAAARCH) was created by the IRTF to take a wider look at AAA issues and architectures. The AAAARCH has been coordinated closely with the AAA WG to define the next-generation AAA architecture. Nowadays, their immediate goal is the definition and detailed description of the accounting operational model for each type of network access.

Another organization, IPDR is an open consortium of leading service providers, equipment vendors, system integrators, and billing and mediation vendors. IPDR supports system interoperability and aims to reduce integration time and cost. Their standardization activity focuses on the definition of a usage record format and the specification of delivery protocols. Since 1999, IPDR has delivered the Network Data Management-Usage (NDM-U) specification [C02], which has been tested by IPDR providers and vendors step by step. Nowadays, NDM-U version 3 for IP-based services has been specified. This initiative emerges for facing the most common of today's standards for capturing and transferring accounting and billing information (e.g. CDR). They are not sufficient for next generation services. They are either insufficiently flexible to reliably represent emerging services or are proprietary to specific vendors.

The following paragraphs describe some standard accounting protocols that can be considered in the scope of the Amigo project [A01] [M01]:

- The RADIUS protocol [R00] has been used for the exchange of authentication, authorization, and configuration information between network access servers and other network elements. The extension of RADIUS in order to also carry accounting information was achieved by defining an appropriate set of accounting attributes (e.g. Acct-Session-Id, Acct-Input-Packets). RADIUS is a widely deployed protocol that may be used for exporting usage information. However, it can only handle a few outstanding requests and has extensibility challenges due to its limited command and attribute address space.
- The Diameter protocol is a framework to define a policy protocol for AAA servers and resource control. It follows the basic RADIUS model but allows a single server to handle policies for many services. The extended Diameter protocol could be considered appropriate for securely transferring accounting records over the Diameter base protocol.
- The SNMP protocol has been widely deployed in a variety of inter-domain accounting applications. With reference to authentication and authorization, many important issues with previous versions of SNMP have been corrected in the SNMPv3 specification. Although there may be some utility in its use for accounting, SNMP is not generally acceptable as a general AAA protocol.
- The IPDR streaming protocol is designed to be a reliable, fast, efficient and flexible accounting protocol. The protocol is developed to address the critical needs for exporting a high volume of data records from the service element with efficient use of network, storage, and processing resources.

It is expected that the Amigo system will promote a new market for home applications and services. New business models will arise with a large potential for service providers and operators. In order to support these business models, the middleware architecture needs to incorporate accounting and billing functionality.

References:

[AAH00] Aboba, B., Arkko, J., Harrington, D., "Introduction to Accounting Management", RFC 2975, October 2000.

[A00] Aboba, B. et al., "Criteria for Evaluating AAA Protocols for Network Access", RFC 2989, November 2000.

[BB00] Brownlee, N., Blount, A., "Accounting Attributes and Record Formats", RFC 2924, September 2000.

[C02] Cotton, S. et al., “IPDR Standards, Network Data Management Usage (NDM-U) Specification“, Version 3.1.1, http://www.ipdr.org/documents/NDM-U_3.1.1.pdf, 2002

[FDL00] Falkner, M., Devetsikiotis, M., Lambadaris, I., “An Overview of Pricing Concepts for Broadband IP Networks“, IEEE Communications, 2000.

[M01] Mitton, D. et al., “Authentication, Authorization, and Accounting: Protocol Evaluation”, RFC 3127, June 2001.

[R00] Rigney, C., “RADIUS Accounting”, RFC 2866, June 2000.

4 Intelligent user services

Intelligent user services can be understood to be services that make the system “intelligent” from the end-user viewpoint. To make the system (home environment) much more attractive to the user, intelligent services among others would include context awareness (§4.1), natural multi-modal user interfaces (§4.2) and user preference modelling and profiling (§4.3). These technological components are overviewed in this chapter.

4.1 Context management

Intuitively, many people perceive context as an aspect from the users’ environment like location and temperature. Despite this common notion, it is hard to define context precisely. The term ‘context' is overloaded with a wide variety of meanings, depending on the purposes of the particular application and/or on the research community standpoint [TACS98].

Several research communities like Information bases, artificial intelligence, human computer interaction and ubiquitous computing, have proposed definitions of context and context awareness. We adopt general definitions proposed by Dey [DSA01]:

Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves[4].

Context-awareness is a property of a system that uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task.

Different types of context can be distinguished. For instance, Schilit defines three categories of context [SAW94]:

- Device context: contextual information related to devices. Examples are available memory, computation power, networks (and their quality), codecs etc.
- User context: context information that describes an individual.
    i. Personal context: health, mood, schedule, activity etc.
    ii. Application context: email received, websites visited, preferences etc.
    iii. Social contexts: group activity, social relationship, people nearby etc.
- Physical context: contextual information related to the physical environment of an entity (device, room, building, and user). Examples are location, time, weather, altitude, and light.

This list is not a systematic approach but merely gives a classification of context by means of examples. It is also clear from this list that not all types of contextual information can be easily sensed. Some types of contextual information (e.g. the mood or activity of individuals) can only be derived by intelligent combination of other information, or by human inputs.

Context management is the functionality that is responsible for propagating context information from its sensor to the application, storing it, controlling possible manipulations on it (e.g., aggregation and the inference of new information), and providing access control to the context information. Generic context management environments exist in the ubiquitous communications world, e.g. [GS01] [D00].

An important aspect of context management is the exchange of context information. Depending on the architecture of the underlying network and the overall context management system different approaches exist, ranging from completely centralized (all context is stored in a central repository that guarantees consistency and a single view) to completely decentralized (applications maintain the context by themselves without a platform that provides management and exchange functions, leading to inconsistencies). In practice, the architecture of the context-awareness infrastructure is often coupled to the underlying networking infrastructure. In spontaneous networking environments, e.g. when a user leaves the house or is on the move in general, access to one central service is not always a good solution. First, the context information from a large coverage has to be propagated to the infrastructure and secondly mobile users have to obtain it via costly and bandwidth limited communication.

[4] In AMIGO, we extend this definition to include device-to-device communication

A solution typically used in ubiquitous computing is to model 'hot spots' that are covered by a low cost network with distinct quality, such as IEEE 802.11b. Such hot spots can offer basic services and context information for a distinct area, thus building a so-called smart or intelligent environment. In contrast to that one could only rely on ad-hoc networking. In particular when context information is spatially restricted, the context information can be propagated in ad-hoc settings. But in a lot of situations the context information needs to be exchanged between different domains, e.g. a personal device in the personal area network and a home-networked device, or towards an external service provider (e.g. a video on demand service that wants to optimize the quality for its clients). Hence, there is a need for standard exchange mechanisms in this area. Cross-domain context information exchange by nature involves security and privacy concerns with respect to the transferred information.

However, we are not aware of such standards for generic context information. Again, in the area of location information some standardization effort is ongoing: there exist DHCP extensions for location information [RFC3825] that provide hosts with their current geographical location, and extensions to presence systems [JEP-0080] that extend jabber/XMPP [RFC3920]. Standardized presence protocols, either XMPP or SIP [RFC3856], are the likely candidates to transfer rapidly changing attributes (e.g. context information) in a secure manner between entities, also in different domains. Most platforms use proprietary, often HTTP-based, mechanisms for context information exchange. An interesting approach that integrates HTTP transport with XML data modelling and XPATH-like expressions is XCAP [R04], currently under investigation in the IETF SIMPLE group. This could be used for remote application access to well-known context information objects. However, there is no support yet for quality-of-context in each of these standards.

For inference of new types of context information (e.g. a room with light and noise can be considered to be occupied) we need semantic definitions of context information. This is covered in the next section in more detail. There exist a lot of rule-based systems that can be used in this respect, but it depends of course on the modelling of the context information itself. Because location context has been the subject of much development, we have devoted a section on location awareness.

4.1.1 Semantic context modelling

To be able to develop a context-aware application it is necessary to address modelling of the storage and retrieval of contextual information. Storing and retrieving contextual information are functions of a context management system. Current approaches to context management are introduced later in this document. Here, we focus on how context information can be modelled.

Modelling and reasoning about context has been the subject of much debate over the past few decades. The concept of context has been studied in several disciplines including Computer Supported Cooperative Work (CSCW), natural language understanding and artificial intelligence and relatively recently, pervasive computing. Here, we cover the work done in these disciplines.

Context information is central in a context-aware application. It has to be shared by many applications, and has to be understandable, unambiguously and manipulated by various parties within a wide distributed system. For example, an RFID sensor in a room may provide information at several levels and to several applications. The provided information can then be used to indicate the presence of someone or to enable their identification or even to simply know their position. This requires context information to be understood by different applications. Thereby it raises the concern of its meaning.

We need to represent context information in such a way that the meaning of the information we describe is explicit in the representation, and not simply implicitly encoded by the application engine that will interpret this representation. Standard data representation techniques such as those used in databases (p-tuples) or XML schemas, while necessary for syntactical interoperability among machine processing applications, lack semantic ground for expressing the meaning of context.

Early attempts to model the semantics of context have been engaged in artificial intelligence under the viewpoint of model generality. Using the framework of logic based knowledge representation, [C96] [G91] have introduced the notion of context as a means for extending general models of some domains with specific axioms which enable reasoning to potentially increase the number of deductions which could be derived. For example, from the general model of buying/selling transaction, we could define the more limited context of very simple transactions. In this context, we could state the axiom that 'after the sale, the buyer owns the object'. Although this axiom doesn't always hold in the general complex world of financial transactions, in the context of very simple transactions it could help infer more information on the object of the sale ownership than what could be inferred from the general world.

This logical model of context provides a calculus for handling context subsumption, and for reasoning from one context to the other, using a model of context as a first class object and predicates relating contexts to domain objects. Under this approach, we can mention work described in [NP01], as CYC or SUO projects, whose aim is to create a standard ontology for describing the whole world that inevitably includes the description of context. Finally, C-OWL is described in [G03] as contextualized ontologies. Using the OWL-DL language enriched with logical rules, it extends ontologies with the capability of keeping the source and the target ontology of a specific piece of information, to enable their interpretation in a local domain and to map two elements of two ontologies that are contextually related. More generally, this could be viewed as attempts to make knowledge/ontologies more modular.

In contrast to modelling context intentionally (as a first class object), modelling context by extension, i.e., by enumerating what is true in a context by definition, seems a more tractable way of modelling context. This approach has been addressed by deKleer in [K86], but with the restricted scope of hypothetical reasoning where context represents alternative variants of an incompletely known world. Closer to the domain of pervasive and ubiquitous computing, this approach has been adopted as well.

To begin with representations that handle the context in a wide sense, let's talk about models that do not use semantic Web technologies. Dey in [D01] proposes a toolkit to handle context information, in a way that is independent from its source, and to propagate it throughout the whole system. This representation of context is application specific, i.e., the programmer providing the code to process the context information determines its format. Another model proposed by Henricksen in [HIR02] is based on three entities that are person, device and channel. It handles the location of these entities, activity for the person, and the quality of the information. Because it is an object-based model, all services entering the system must know the data model and all methods to manipulate it. As with Dey’s model, this model can handle various kinds of context information but doesn't allow its sharing in an easy way. For those two models, the representation of context information is not semantic, and cannot be understood except by applications that have foreseen manipulating it and integrated specific code to do so. A distribution model is under study to complete Henricksen's model. On the other hand, there are few models for context representation that use semantic representation. Chen in [CFJ03] proposes the CoBra ontology to represent the context with OWL tools. In its version v0.2, it contains more than 100 concepts to describe a university context with the agent, the place, the activity and the location. This model does not offer an architecture to exploit it yet, but is under development.

Current work on the extensional modelling of context is favouring the use of semantic Web languages such as RDF and OWL to model a context as a set of instances derived from a generic object based representation of the domain. Such languages have been introduced to model information in such a way that it could be shared, and understood without external knowledge and reasoning. Concerning 'physical world' models, the Nexus project proposes the AWML (Augmented World Modelling Language) to model real-world objects and their context as attributes and an augmented world like the Web, a data-base or other informatics artefacts. It also provides an architecture to manage augmented world information with a query language and a description language. PML (Physical Markup Language) is an XML-based language like AWML. It focuses on the representation of everyday objects and their state in a standard way. In work conducted by [P01], a work office environment is modelled as a tree based on the analysis of differences and similarities between physical and virtual environments. The concept of Physical-Virtual Artefacts (PVA) is introduced and represents both the virtual environment offered by computer systems and the surrounding physical environment of the user. Then, the user defines three-dimensional spaces in the physical environment called 'active volumes' and puts their PVA in relation with 'active volumes' in a tree like hierarchy. This model allows users to control and configure the world representation. This model is oriented for object modelling and location.

Finally, Flury in [FPR04] propose a location model using semantic Web technologies. Their model can support location process using four different models (semantic model, graph theory model, set-based model and geometric model) expressed with OWL.

Concluding, context modelling is a novel area of research still under debate. Context is difficult to handle because of its ambiguous nature. We need a model to define its semantics. Furthermore, to be able to do context reasoning, for instance to infer higher level context, we also need a clear and unambiguous context model. As indicated, currently some interesting context modelling approaches are researched which, when matured, may provide a mechanism to effectively model context.

4.1.2 Location awareness

As an objectively measurable physical value, location is as a major component of context information. In the past decade, research has focused on location-sensing technologies, location information representation, management, delivery and location-based applications [HSK04]. These domains are the basis of location-aware computing.

Location sensing technologies

Large-scale outdoor location-sensing has become commonplace with GNSS (Global Navigation Satellite System), which include GPS [G93] EGNOS, GLONAS and GALILEO.

Location determination with cellular terrestrial mobile telephone networks (which include GSM and UMTS) is also commonplace using such techniques as Cell-ID, EOTD, TDOA, etc.

Assisted GSNSS (with cellular terrestrial networks) complementing information obtained from satellites with information sent by the terrestrial network, may be used to improve on GNSS systems and possibly extend their use to indoor environments.

Indoor location information has been tested first with such special-purpose contraptions as the active badge system [WHF92]. With this system, users wear badges that emit infrared signals received by sensors distributed in a building. This system can provide discrete information about the presence of a user inside a particular room.

The major shift with today's location-sensing technologies is the use of systems not originally dedicated to acquire location information. The user does not have to perform any explicit action to be located or to wear specific equipment; location systems rely on technologies already used for identification, communication and so on.

The first example of this shift is the widespread use of RFID (Radio Frequency Identification) tags, a cheap, contactless, passive identification technology [WFG99]. It is primarily designed to identify people or equipment, and to track and manage goods. An antenna uses inductive coupling to supply power to the tag, it can then write or read data from it. These passive tags are a cheap solution to locate a tagged object within the range of the antenna. For example, it can detect the position of a book on a shelf, the entrance of a user inside a room and so on. Adding onboard battery to RFID tags can extend their range and allow a continuous geometric positioning in space [NLL03].

Another common example of new location-sensing technologies is the reuse of WLANS or WPANS to locate people and devices. In wireless local area networks such as WiFi, the signal strength from access points can be used to estimate the position of a WiFi-equipped device (using a model of the signal propagation or a table of measured signal strength combined with Bayesian filters) as described by Paramvir Bahl [BP00].

Video cameras can also be used to track mobile shapes within their vision range. With the proper homography, the system can calculate the position of the shape within real space.

Location information representation and management

Cartography has been one of the first domains concerned with the representation and management of location information within an information system. The field of geographic information systems provided a database-centric solution [DS96]. The database manages the matching between the geometrical/geographical description of an element and its semantic meaning (according to a domain-specific interpretation of space).

Beyond this kind of proprietary representation bound to a particular database, the Open GIS consortium tried to promote an open recommendation for location information representation based on XML. The GML (Geography Markup Language) is an encoding for the management of geographic information, including both the geometry and properties of geographic features [GML04].

In addition to the representation and management of static information about the environment, location-aware computing uses the dynamic data provided by location-sensing technologies. As these sensors may be multiple and heterogeneous, a direction of research has focused on numeric sensor data fusion. Jeffrey Hightower has recently described a particular filter to fuse the raw numeric location data provided by different sensors [HB04].

Another direction of research has tackled the aggregation of location information at a more abstract level. The research of Privat et al. [PF03] starts from the hypothesis that location sensing technologies and the corresponding representations of location information can be abstracted within a limited set of models that provide a common ground for heterogeneous technologies and make it possible to meet the requirements of location-aware applications.

Abstracting generic location models can go one step further. Using a formal and comprehensive ontology-based modelling of location makes it possible to integrate consistent location information from sparse and heterogeneous data. The work of Flury et al [FPR04] is to design this ontology and to add inference rules to automate the process of reusing the location information (interpretation and aggregation of raw data from the sensors and responding to requests from client location-aware applications).

Platforms for location-based systems exist, an example, is the WASP platform that support location-based directory services, using semantic Web type technology [WASP] [DPSR04).

Location-aware applications

Location-aware systems use location information to offer specific location-centric services or to adapt other services based on this location information.

Location has been widely used for vehicle navigation and guidance, and navigation is the basic application provided with GNSS or AGNSS receivers, whether they are stand-alone or work with a PDA.

Mobile terrestrial cellular networks use physical location for their own needs (handover between cells) but they do also offer this information for either B2B or B2C services. The Enhanced 112 is a set of rules set by the EU to locate and identify a user for emergency calls. Cellular networks also offer other such services such as finding close points of interest (restaurant, theatre and so on), notification of proximity of friends (buddy-lists) or the possibility to map out a route.

Location-aware services also begin to appear in indoor environments (especially when the user carries a digital assistant). Examples are indoor navigation (basically the same as outdoor vehicle navigation but inside a building) or guidance into museums and exhibition halls. Other more industrial services will come along with inventory management systems. The work of Chraiet et al. [BCDEFPV03] goes further and proposes to integrate navigation with other position-based services in an indoor environment. These services include the dynamic discovery of devices and services around the user (using the physical proximity between the user and the device), interface export and import (such as audio/video stream, graphical interface) using the most convenient devices (a TV screen used instead of the digital assistant screen, a home cinema system etc.) or even follow-me interfaces where services use the closest interface device wherever the user moves.

Context management is a key enabling technology for Amigo. Intelligent services and applications need support for this, and therefore a generic context management framework is a required feature of the Amigo middleware. Such a framework is not available yet. There are architectures that cover parts of the puzzle, and (primarily academic) approaches to provide infrastructural support for particular types of context awareness. More concrete scenarios and user requirements are needed to assess whether these architectures can be extended or reused in the scope of the Amigo project. Due to the lack of standardization, and private data formats used in these platforms, interoperability between different approaches cannot be achieved. Currently, standardization is progressing primarily in the area of location-based systems, which is only a part of the context puzzle.

It is a key challenge for Amigo to design and implement a context-management infrastructure that can cope with several types of context information, and provide secure exchange of this information while protecting the privacy of end-users. Such an infrastructure necessarily must be open, in the sense that it is possible to exchange or replicate contextual information with other entities (e.g. exchange room-area information with personal-area information, and personal information with global services).

References

[DSA01] Dey, A. K., Salber, D., Abowd, G. D., “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications”. Human-ComputerInteraction, 16, pp. 97-166, 2001.

[SAW94] Schilit, B., Adams, N., Want, R., “Context-Aware Computing Applications”, IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, USA, 1994

[TACS98] Theodorakis, M., Analyti, A., Constantopoulos, P.; Spyratos, N., “Context in information bases“, Third IFCIS Conference on Cooperative Information Systems (CoopIS'98), New York, New York, 1998.

[DPSR04] Dockhorn Costa, P, Ferreira Pires, L., van Sinderen, M., Rios,D., “Services Platforms for Context-aware Applications”, proceedings EUSAI, Eindhoven, November, 2004.

[PZB04] Peddemors, A., Zandbelt, H., Bargh, M., “A mechanism for Host Mobility Management supporting Application Awareness”, proceedings of Mobisys, Boston, Massachusetts, May, 2004

[GS01] Gray, P., Salber, D., “Modelling and using sensed context information in the design of interactive applications”, proceedings of the 8th ifip working conference on engineering for human-computer interaction (ehci), Toronto, Canada,May, 2001.

[D00] Dey, A, “Providing Architectural Support for Building Context-Aware Applications”, PhD thesis, College of Computing, Georgia Institute of Technology, December, 2000

[RFC3825] J. Polk et al, “Dynamic Host Configuration Protocol Option for Coordinate-based Location Configuration Information”, RFC3825, July, 2004

[JEP-0080] “User Geolocation”, Jabber Enhancement Proposal, http://www.jabber.org/jeps/jep-0080.html, Jabber Software Foundation, 2004

[RFC3920] P. Saint André (ed), “Extensible Messaging and Presence Protocol (XMPP)”, RFC 3920, October, 2004

[RFC3856] J. Rosenberg, “A Presence Event Package for the Session Initiation Protocol”, RFC3856, August, 2004

[R04] Rosenberg, J. “An XML configuration access protocol (XCAP)”, http://www.jdrosen.net/simple_acap.html , internet draft http://www.ietf.org/internet-drafts/draft-ietf-simple-xcap-04 , 2004

[CFJ03] Chen, H.; Finin, T., Joshi, A., “The Role of the Semantic Web in pervasive Context-Aware Systems”, Proceedings of ISWC, 2003

[G03] Giunchiglia, K., “C-OWL: Contextualizing Ontologies”, Proceedings of the 2nd International Semantic Web Conference (ISWC'03), October, 2003.

[K86] deKleer, J., “An assumption-based truth maintenance system”, Artificial Intelligence, 28(2):127--162, 1986.

[D01] Dey, A. K., “Understanding and Using Context”, Personal and Ubiquitous Computing, vol. 5, no. 1, pp 4-7, 2001

[G91] Guha, R., V., “Contexts: A Formalization and Some Applications”, PhD thesis, MCC Technical Report ACT-CYC-423-91, Stanford,USA, December, 1991.

[HIR02] Henriksen, K., Indulska, J., Rakotonirainy, A., “Modelling Context Information in Pervasive Computing Systems”, Pervasive’02, LNCS 2414, pp. 167-180, Springer-Verlag, 2002.

[C96] Mc Carthy, J., “A Logical AI approach to context”, Unpublished note, February, 1996, http://www-formal.stanford.edu/jmc/logical.html

[NP01] Niles, I., Pease, A., “Towards a Standard Upper Ontology”, Proceedings of the international conference on Formal Ontology in Information Systems, Ogunquit, Maine, USA, 2001

[P01] Pederson, T., “Object Location Modelling in Office Environments – First Step”, Workshop on Location Modeling for Ubiquitous Computing, UBICOMP, Atlanta, USA, September, 2001.

[BP00] Bahl, P., Padmanabhan V. N., “RADAR: an in-building RF-based user location and tracking system”, Proceedings of INFOCOM, pp. 775 – 784, March, 2000.

[BCDEFPV03] Blache, F., Chraiet, N., Daroux, O., Evennou, F., Flury, T., Privat, G., Viboud, J-P., ”Position-Based Interaction for Indoor Ambient Intelligence Environments”, Ambient Intelligence, pp. 192-207, November, 2003.

[DS96] Denègre, J., Salgé, F., ”Les systèmes d'information géographique”, Presses Universitaires de France. Paris, France, 1996.

[FPR04] Flury, T., Privat, G., Ramparany, F., “OWL-based location ontology for context-aware services”, Proceedings of the Workshop Artificial Intelligence in Mobile System (AIMS) in conjonction with Ubicomp, Nottingham, UK, September, 2004.

[G93] Getting, I. A., “The Global Positioning System”, IEEE Spectrum. 30 (12), pp. 36-38, 43-47, December 1993.

[HSK04] Hazas, M., Scott, J., Krumm, J., “Location-Aware Computing Comes of Age”, IEEE Internet Computing, pp. 95-97, February 2004.

[HB04] Hightower, J., Borriello, G., “Particle Filters for Location Estimation in Ubiquitous Computing: A Case Study”, Proceedings of Ubicomp, pp. 88-106, September 2004.

[NLLP03] Ni, L.M., Liu, Y., Lau, Y. C., Patil, A.P., “LANDMARC: indoor location sensing using active RFID”, Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 407 – 415, March, 2003.

[GML04] Open GIS Consortium, Inc, “OpenGIS Geography Markup Language (GML) Implementation Specification”, OpenGIS Recommendation paper, February 2004.

[PF03] Privat, G., Flury, T., “An infrastructure template for scalable location-based services”, Proceedings of SOC 2003, pp. 214-217, May, 2003.

[WFG99] Want, R., Fishkin, K. P., Gujar, A.; Harrison, B. L., “Bridging physical and virtual worlds with electronic tags”, Proceedings of the SIGCHI conference on Human factors in computing systems pp. 370 – 377, 1999.

[WHFG92] Want, R., Hoppe,r A., Falcao, V., Gibbons, J., 1992. The Active Badge Location System, ACM Transactions on Information Systems, 10 (1), January 1992, pp. 92-102.

[WASP] The WASP project, http://wasp.freeband.nl

4.2 Multimodal user interactions

Through analogy with person-to-person communication, we propose next to define and classify person-to-machine interaction modalities according to their relation with our five senses or their equivalent in state-of-the-art technologies. These relationships are summarized in the next Figure 4.1. Note that the typology of interaction modalities presented here is not definitive and is only meant to illustrate a possible clustering, which can obviously be adapted and extended.

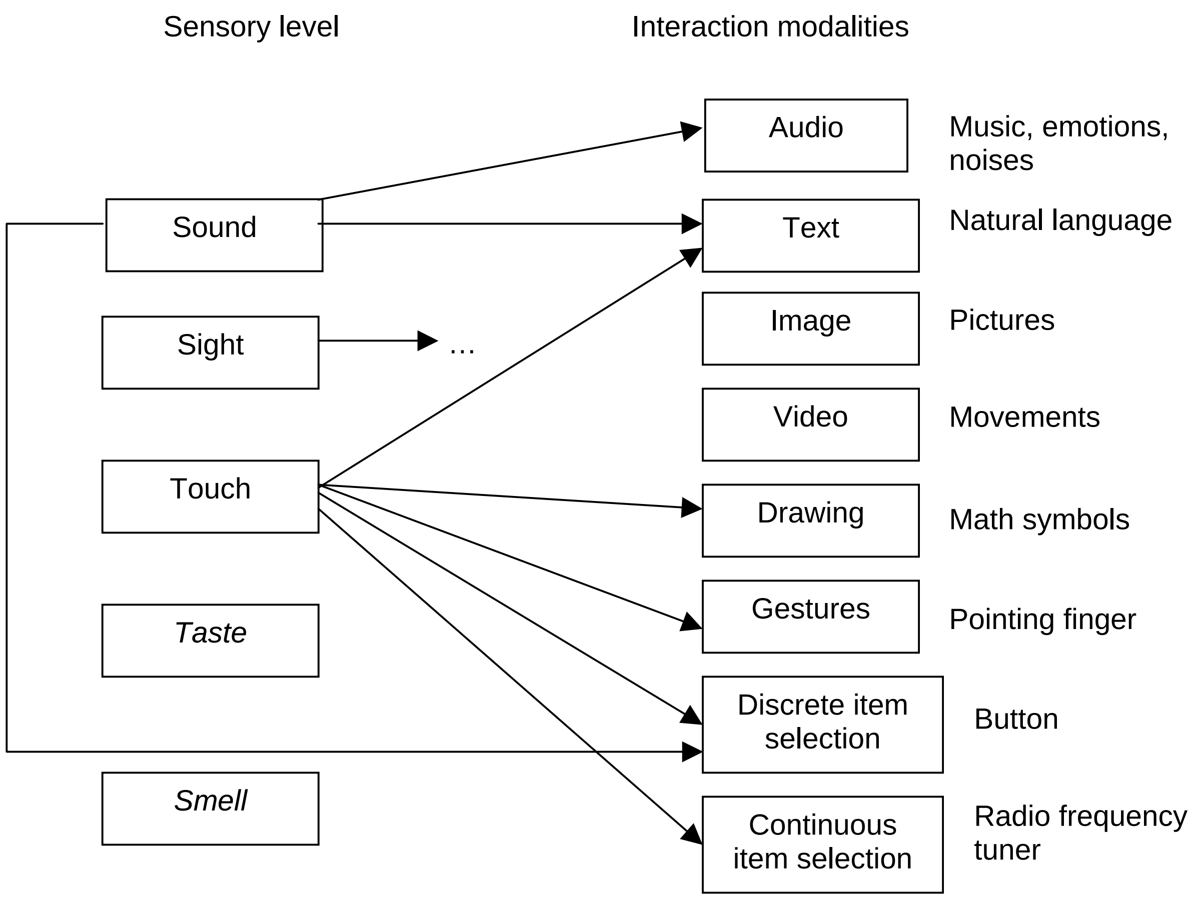

Figure 4.1 Person-to-machine interaction modalities

Some senses are poorly exploited for now by state-of-the-art HCI systems, for example taste and smell, whereas other senses are very rich, like sight and touch. In the diagram, the arrows from sight are not shown, because they actually target every interaction modality except the first one. Let us now give some concrete examples for every possible sense and associated modality:

Sound

- Any type of audible sound may be used for direct or implicit interactions. Imagine for example a scenario where James Bond asks his computer to fire a gun when he will whistle Bach's Chaccone. Many other examples (more realistic but less amusing) of audible interactions exist.
- Large vocabulary speech recognition can also be used to transfer some spoken text or discourse to the dialogue manager for interpretation.
- Small vocabulary speech recognition or word spotting can also be used to select some item, like in command-and-control applications for example.

Sight

Vision offers the widest range of possibilities concerning user interactions. The following algorithms can be used for example:

- Optical Character Recognition to recognize some text written by the user.
- Image recognition can be exploited to find out some item that resembles the picture drawn by the user.
- User and face tracking algorithms can be used to localize the user.
- Diagrams drawn by the user can be recognized and interpreted.

- 2D or 3D gestures made by the user can be interpreted for example to select or move virtual objects in augmented reality applications.
- Target items can be selected using pointing gestures.
- Continuous values can also be exploited, for example when using physical captors, like an eye tracker that gives the instantaneous value of the 3D angle that the user is looking at.

Touch

By analogy with the human hand, we can encompass within the “tactile” sense the following kinds of information:

- Pressure level
- Temperature level
- Recognition of surface textures

Therefore, every device that exploits one of these kinds of information can be classified as tactile. In our view, this includes keyboards, pressure and temperature captors, mice, haptic devices etc. From this point of view, tactile senses are already largely exploited by traditional user interfaces. They are used for example to:

- Input some text in natural language with some keyboard.
- Draw some diagrams on a touch-screen.
- Analyze and interpret 2D gestures realized with an electronic pen on a touch-screen, such as scribbling, circling some important item etc. The meaning of such interaction gestures differs from the meaning of drawings as mentioned previously.
- Press a button. This also differs from item selection using sight, where physical pressure information is not exploited at all, but the computer uses a camera to “recognize” a pointing gesture and estimate the position of the hand.
- Capture some analogical information from a temperature captor for example.

Note that to the best of our knowledge, state-of-the-art HCI systems do not yet use surface textures as an input modality.

Although sight and touch can be exploited from many perspectives, speech is the *Grail* of advanced interfaces for several reasons:

- It carries text, on which our languages are based and languages are the most accomplished ways of communication between humans, because of their tremendous capability to transmit complex information compared to other modalities.
- It is the fastest solution to transfer complex sentences and ideas.
- It is the most natural modality of interaction between humans, and historically the first one.

This is why speech interactions are treated separately in a following section.

Combining modalities

Combining modalities for user interactions can serve several purposes:

- Disambiguate the user’s inputs: for example, speech recognition and lip-reading using cameras can be combined to improve the accuracy of speech recognition [W93]. More generally, Oviatt showed that 80 % error avoidance could be achieved via mutual disambiguation across media [O99].
- Accommodate a wider range of users than traditional interfaces: multimodal interfaces especially provide a clear added-value for disabled and handicapped people, but also for specific categories of people [O03].

- Capture and merge different kinds of information that might be important to interpret and understand the user's inputs. For example, speech recognition can be combined with facial expression recognition in order to label the text stream with meta-data such as sentence types (questions, orders etc.), dialog acts, emotions etc.
- Improve the naturalness of ambient intelligence user interactions, as real communication between people naturally exploits several modalities [N03].
- Improve the effectiveness and quality of the way information and feedbacks are presented for the user, for example with an "emotional" talking head.

Implicit interactions

Most person-to-person interactions are implicit. For example, looking at someone else may have different meanings depending on the context and "shared knowledge". This notion of "shared knowledge" is central to any communication between two entities [RVDA05]. It includes a common language and many notions that are shared by the participants. Implicit interactions heavily rely on this shared knowledge, which shall be trained automatically or defined manually in every ambient intelligence system. It can be as simple as a garage door that "knows" it should open when the user's car is approaching, or as complex as the implicit meaning of someone's look. Ideally, this common knowledge shall include all the concepts of our human world and history, but our state-of-the-art technology can only handle for now very restricted concepts constrained to a specific application domain. An interesting perspective of ambient intelligence user interactions consists of automatically training such a common knowledge by observing our world and how we (humans) interact with it and between us.

State-of-the-art multimodal platforms

Many experimental platforms and systems have been developed over the past few years that include some form of multimodal interaction. Most of them use speech input combined with pointing gestures. We present here a short list of such recent state-of-the-art systems, which are representative of different views and concepts about multimodal user interactions:

- One option for experimenting with multimodal user interactions is to completely submerge the user in a virtual world, which is totally under the control of the computer. Such systems are now called CAVEs, and the first one has been built in 1992. Their main advantage is that the "common knowledge" (as defined above) between the users and the system is well-defined, because the virtual world is defined and controlled by the system. For example, a virtual music centre is presented in [NH00].
- The previous option has an important limit: the submerged user can not interact any more with the "real" world. Therefore, this is the current trend of ambient intelligence, rather than focusing on adding practically invisible interfaces onto the real world. For example, Robert Krivacic developed pads, which are intermediate devices between a laptop and a sheet of paper, and can be used as mobile touch-screens.
- The tangible bits project [IU97]: Hiroshi Hishii (MIT Media Lab) conceived a room, where the user interacts with the system by moving physical objects. On the other hand, the system outputs information by projecting lights onto the walls of the room.
- Bodyarchitecture [C05] is an artistic project that analyses the movements and sounds of the user to alter and deform their surrounding visual and audible environment.
- The Dove system [OFCS03] supports collaborative interactions between distant persons through video, speech and gestures analysis via an electronic pen and a tablet PC.
- The AdApt system [GBBB02] aims at helping the user to understand and have an efficient feedback when using a multimodal real-estate application. The constraints of the user are represented by icons that can be manipulated on a touch screen along with speech inputs.

- The EMBASSI project proposed a modal XML-based UI description language [Dubinko03] to transfer information to mobile devices using different modalities (for example in Braille for disabled users) [HKSE01] [R01].
- The SmartKOM system (http://www.smartkom.org) developed speech and pointing with hands in augmented reality.
- The Quickset architecture merges 2D gestures with speech input to handle map-based applications [CJMO97]. A number of other projects, such as the *OZONE* project, which is described in detail in the following sections, use the same concepts to enhance specific parts of this multimodal dialogue architecture.
- The MIAMM project (http://www.miamm.org) develops a portable device that merges speech recognition and haptic interactions for a music selection application.
- The Multimodal Interaction working group of the W3C has been created in 2002, with the objective of “extending the Web to allow users to dynamically select the most appropriate mode of interaction for their current needs”. Until now, this group has focused on the use cases and interaction requirements, and is now (2005) working on an extensible multi-modal annotations (EMMA) language and on the definition of a multimodal architecture.

References:

[C05] Cantoni, R., “Bodyarchitecture: the evolution of interface towards Ambient Intelligence”, in Ambient Intelligence, eds. G. Riva et al., IOS Press, 2005.

[CJMO97] Cohen, P.R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., and Clow, J. QuickSet, “Multimodal interaction for distributed applications”, In Proceedings of the Fifth ACM International Multimedia Conference ACM, NY, 31–40, 1997.

[DKMR03] Dubinko, M., Klotz, L.L., Merrick, R., Raman, T.V., eds., “XForms”, 1.0 W3C Recommendation, World Wide Web Consortium, 2003.

[GBBB00] Gustafson, J., Bell, L., Beskow, J., Boye, J., Carlson, R., Edlund, J., Granström, B., House, D., Wirén, M., “AdApt – a multimodal conversational dialogue system in an apartment domain”, Proc. ICSLP, 2000.

[HKSE01] Herfet, T., Kirste, T., Schnaider, M.: Embassi, “Multimodal assistance for infotainment and service infrastructures”, Computers & Graphics vol. 25, pp. 581–592, 2001.

[IU97] Ishii, H., Ullmer, B., “Tangible bits: towards seamless interfaces between people, bits and atoms”. Proc. of CHI, ACM Press, pages 234-241, March, 1997.

[NH00] Nijholt, A., Hulstijn, J., “Multimodal Interactions with Agents in Virtual Worlds”, Chapter 8 in Future Directions for Intelligent Information Systems and Information Science, N. Kasabov (ed.), Physica-Verlag, Springer, Heidelberg, pp.148-173, 2000.

[N03] Nijholt, A., “Algorithms and Ambient Intelligence”, W.F.J. Verhaegh, E.H.L. Aarts & J. Korst (eds.), Kluwer Academic Publishers, Boston/Dordrecht, London, 2003.

[OFCS03] Ou, J., Fussell, S.R., Chen X., Setlock, L.D., Yang, J., “Gestural communication over video stream: supporting multimodal interaction for remote collaborative physical tasks”, Proc ICMI, Canada, 2003.

[O99] Oviatt, S. L., "Mutual Disambiguation of Recognition Errors in a Multimodal Architecture", ACM Conference on Human Factors in Computing Systems, Pittsburgh, PA, May 15-20, pp. 576-583, 1999.

[O03] Oviatt, S. L., "Multimodal interfaces”, in Jacko, J.A., Sears, A., eds: The Human-Computer Interaction handbook, 2003.

[R01] Richter, K., “Remote access to public kiosk systems”, In Proceedings of the 1st International Conference on Universal Access in Human-Computer Interaction, 2001.

[RVDA05] Riva, G., Vatalaro, F., Davide, F., Alcañiz, M., (Eds.), “Ambient Intelligence”, IOS Press, 2005, http://www.ambientintelligence.org

[W93] Waibel, A., “Multimodal Human Computer-Interaction”, Proc. Eurospeech, 1993.

4.2.1 Speech processing

As already stated, speech interaction is only one possible interaction modality in current day systems. Nevertheless, its main advantage over the other modalities is that it can easily carry extremely complex information/interaction via natural language. It is thus a very rich way of communication between humans, and even our most advanced research platforms are far from exploiting all of its potential.

A speech-based dialogue system consists of a number of components: the system starts by receiving a speech signal from the user as input, recognizes the spoken words, understands the meaning of the utterance and at the end produces speech output that expresses the system's response to that input. It has been working in four different aspects: signal pre-processing, speech input, speech output and speaker recognition.

Signal preprocessing integrates various techniques for capturing speech input and ensuring that the signal is received as clearly as possible and thus enhancing the performance of speech recognition. The following modules can be identified:

- Array management: in the home/office environment multiple microphones or microphone arrays are required in order to cover the entire area optimally. Depending on the user's position the array management module decides on using the combination of microphones with the best signal.
- Beamforming: this is responsible for speech acquisition by restricting the receptive field to the desired talker with the objective of signal separation and ambient noise suppression.
- Noise cancellation and room acoustic suppression: in most environments, there is a high level of noise that must be filtered before processing the signal (i.e. in a normal home or mobile environment it is very likely to have radios and TV noise, side conversations or environmental noise). These modules reduce extraneous voices by using various acoustic echo-filtering techniques.

Speech input is the unit where the system recognizes the user's speech understands the meaning of the spoken utterance and acts accordingly. Three main modules are:

- Automatic speech recognition: speech recognition is the process of converting an acoustic signal to a set of words. The conversion is made by comparing the features of the signal to three sources of knowledge: an inventory of acoustic reference models (Acoustic Models), a list of words that the system can recognize (Lexicon) and information about the likelihood of particular word sequences occurring in a given context (Language Model).
- Speech understanding: the output of the speech recognition module is a set of hypotheses as to the correct transcription of the spoken utterance. These hypotheses form the input to the understanding module, which interprets the utterance on the basis of language rules (Grammar), specifically compiled for the application in question, and generates the semantic output of the spoken utterance.
- Dialogue handling: during natural dialogue interaction the system takes into account the semantics of the user's input, what it knows it can do (System Knowledge) and what the system knows about the user (User Model). The dialogue-handling module (or Dialogue Manager) combines this information to decide the system's response or action to the user's input.

Speech output is the unit where the system generates and presents to the user its response. Depending on the current status of the interaction, this response may be a question, a confirmation or the information requested by the user. Two main components are identified:

- Natural language generation: language generation is the process of preparing the output speech in line with any rules on what would be appropriate, putting it in the relevant context so as to make it easier for the user to understand it and forming the output utterances following grammatical rules.
- Text-to-speech synthesis: in simple dialogue systems pre-recorded messages are used for system output. In more advanced systems the output of the language generation is fed to the speech synthesis engine that dynamically converts the written utterance to a speech signal to be played by the system.

Speaker recognition is the process of automatically recognizing who is speaking on the basis of individual information included in speech waves. This technique makes it possible to use the speaker's voice to verify their identity and control access to services and information.

Speaker recognition methods can be divided into speaker identification and speaker verification. **Speaker identification** is the process of determining which registered speaker provides a given utterance based on the comparison of the user's voiceprint with the set stored in the system's database. **Speaker verification** is the process of accepting or rejecting the identity claim of a speaker.

One important user requirement is that most users want to control the ambient Intelligent system. This implies that explicit as well as implicit user interactions shall be supported with the highest quality possible. To control complex functions such as the ones proposed in ambient intelligence services, complex interactions are required as well. Moreover, in changing and mobile environments this requires the use of speech interaction and natural language processing. This is not in contradiction with the new implicit interactions/invisible interfaces paradigm, which is one of the main objectives of ambient intelligence platforms, but both explicit and implicit interactions are complementary and shall be supported. Furthermore, as shown before, implicit interactions are based on an important context model (world model or “shared knowledge”), which is constantly updated by the system observing the environment and the users. This implies that the system tracks and interprets the user’s activities, which also requires listening to the users when they are talking together, or even to “understand” the radio/TV news when the user is listening to them. As we can see, speech interpretation and understanding is extremely important for ambient intelligence platforms, for explicit user interactions, and even when there is no direct interaction at all.

In regards to Amigo:

First, the naturalness of user interactions shall clearly be improved, as nowadays technologies do not support adequate features to capture and interpret natural communications between and with humans. This includes developing efficient techniques to support multimodal speech and 2D and 3D gesture recognition, and to improve the overall processing chain from speech acquisition to natural language understanding.

Second, ambient intelligence platforms follow the user when they move, and this mobility creates specific challenges to be addressed. One of the advantage of speech interaction is that the sensors to capture audio are very small and cheap, and can easily be multiplied and put everywhere, even on the user them self. Technological challenges include supporting and efficiently handling a variety of environments and a multiplicity of microphones and recording conditions, to distribute the required processing power over the network and capable nodes, which requires designing modular speech processing elements and handling the distortion resulting from the networks and the compression tools used to transmit the audio stream, and to provide standards to enable the rapid and effortless development of new services.

Third, efforts have to be realized to support and enhance the emergent paradigm of implicit and invisible interactions. This includes extending technology from explicit to implicit interactions, which implies (semi-) automatically building context models by tracking, observing and interpreting the user’s behaviour. This is actually much more difficult to achieve than traditional direct interactions, where the user is consciously talking to a computer system, and makes cognitive efforts to clarify their inputs. This challenge can only be addressed for now in a limited and well-defined application domain.

4.3 User modelling and profiling

User modelling is a very broad research area [R89] [KW95] [LJS99 [BGV01] [03] [F01] [Z03] with decades of historical development. In general, the concept of user modelling (also known as personalization, user profiling, or adaptive user interfaces) addresses issues of understanding users in order to make a system useful and make user-system interactions user friendly and universal. We may distinguish between at least the following concepts of user modelling:

- In design-time adaptation to users, we employ various methods to anticipate users’ needs, goals, interactive behaviour etc. in order to proactively fit the system to its future users. The system may be programmed at design time to dynamically respond in different ways depending on the user’s behaviour, but it does not offer the user any other ways of modifying the system’s behaviour.
- The system may be customisable by users who can modify the system’s behaviour in various ways, such as functionally or with respect to its style of interaction, by selecting among customisation options.

- In user model-based adaptation, the system may itself observe the user's behaviour and adaptively modify its interactive behaviour online as a function of the data gathered. Thus, user model-based adaptation is: on-line, made by the system, and uses collected and stored user information.

As a variety of users may operate with the system, a user model is a representation of the properties of a particular user or group of users. More simply, user models serve as a description of the users of a system and a prediction of how they will behave and perform tasks. User models can be categorized according to diverse characteristics:

- On the basis of how many users the model will cover*:* canonical and individual models*.* Canonical models are usually applied in "classical", all-purpose systems (do your best to accommodate everyone). On the other hand, individual models are more flexible, used in individualized interfaces and tailored to the single user. The decision of canonical versus individual user models has a profound impact on the design of the interface. Canonical models are designed once and for all with the system structure while individual models need to be built and maintained for any new user. A common approach is using a default model that contains all known characteristics of the user community and can let the user immediately use the system, then this model can be updated with additional information characterizing each individual user.
- On the basis of how the system will build and maintain the model*:* explicit versus implicit models*.* In the explicit user model approach, much of the information about the user is added by specific actions on the part of system designers or users. The system may allow the users to modify it according to their wishes (e.g. by defining shortcuts and macros of actions), or, it may directly query the users for their preferences. Then, the system classifies the users with respect to a pre-defined set of possibilities (a stereotype). Implicit user models are built by the system on the basis of normal interaction. They can address simple facts (e.g. activating the task the user was working on last time) but also comply to more sophisticated behaviour, like computing presuppositions (e.g. "the user is fast at typing -> expert user"), reasoning about a user's beliefs (e.g. "if user knows X then probably knows Y too"), or inferring a user's plans (e.g. "the user is writing lots of single lines -> is probably willing to create an itemized list").
- On the basis of volatility of the information kept about a user*:* long-term versus short-term models*.* Both long-term and short-term user characteristics include user preferences (e.g. background colours, current printer) and cognitive factors (e.g. user level of mathematical knowledge, current goal). In long-term (or historical adaptable) user models the system collects and stores information on the individual user's behaviour for subsequent interactions. In short-term (or instant adaptable) ones the point is to apply the information gathered as soon as possible, which then may be safely deleted at the end of the interaction. Hybrids between historical adaptation and instant adaptation are possible.

References:

[R89] Rich, E., "Stereotypes and user modelling", User Models in Dialog Systems, Springer, 1989.

[KW95] Kobsa, A., Wahlster, W., "User Models in Dialog Systems", Springer, Berlin, Heidelberg, 1995.

[LJS99] Linton, F., Joy, D., Schafer, H., "Building user and expert models by longterm observation of application usage", In Proceedings of the seventh international conference on User modeling, Springer-Verlag, New York, 1999.

[BGV01] Bauer, M., Gmytrasiewicz, M., Vassileva, J., "User Modeling", In Proceedings of 8th International Conference, Springer-Verlag, Berlin Heidelberg, 2001.

[J03] Jameson, A., "Systems that adapt to their users: An integrative overview", In Tutorial presented at 9th International Conference on User Modelling, Johnstown, PA, USA, 2003.

[F01] Fisher, G., "User Modelling in Human-Computer Interaction", UMUAI 11, pp. 65-86, 2001.

[Z03] Zukerman, I. et al, "Predictive Statistical Models for User Modeling", UMUAI 11, pp. 5-18, 2003.

5 Ambient system architectures

This chapter presents ambient intelligent system architectures developed in the scope of the most famous research projects: namely MIT Oxygen (§5.1), ITEA Ambience (§5.2), and IST Ozone (§5.3).

These reference architectures can be seen as a first step towards the more advanced ambient intelligent architecture targeted in Amigo. The concepts and approaches used in these projects such as human-centred computing, natural interfaces, pervasive services, flexible self-configured networks, powerful, energy-efficient computing, terminal and network QoS, adaptable distributed middleware, pervasive Web services have much in common with Amigo. These concepts, approaches, and solutions are overviewed, analysed and furthermore assessments and challenges for Amigo research and development are identified.

References:

[Oxygen] http://oxygen.lcs.mit.edu/overview.html

[Ambience] http://www.hitech-projects.com/euprojects/ambience/

[Ozone] http://www.hitech-projects.com/euprojects/ozone/public_public_documents.htm

5.1 Oxygen

The Oxygen project that was originally envisioned by Michael Dertouzos is expected to be a collection of technologies embedded in workplaces and homes working seamlessly together and often behind the scenes. Oxygen is a five-year project that started in 2000 with an effort of $50 million dollars, including more than 150 MIT researchers and six leading industrial partners such as Nokia and Hewlett-Packard. Oxygen claims that *Oxygen makes it easier for people to do more by doing less by bringing abundant computation and communication, as pervasive and free as air, naturally into people's lives.* Its manifesto is based on the following new concepts introduced (MIT 2004).

5.1.1 Concepts used

Human-centred computing supports social interaction in its familiar context. People communicate with computation easily, as they do with each other, using shared knowledge and intelligence.

Space-centred computation embedded in ordinary environments defines **intelligent spaces** populated by cameras, microphones, displays, sound output systems, radar systems, wireless networks, and controls for physical entities such as curtains, lighting, door locks, soda dispensers, toll gates, and automobiles. People interact in intelligent spaces naturally, using speech, gesture, drawing, and movement, without necessarily being aware that computation is present.

Person-centred devices provide universal personal appliances that are inexpensive and can be carried and used anywhere. They are equipped with perceptual transducers such as a microphone, speaker, video camera, and display.

Flexible networks connect dynamically changing configurations of self-identifying mobile and stationary devices by integrating different wireless, terrestrial, and satellite networks into one seamless internet through algorithms, protocols, and middleware.

Adaptable software systems can be customized easily for meeting the needs of individual users and the environment and taking advantage of newly published software.

Perceptual interactions consist of spoken interaction and vision interaction. Spoken interaction allows humans to communicate with computers, in much the same way they communicate with one another, using speech to create, access, and manage information and to solve problems. Vision interaction allows humans to interact with computers, much as they interact with one another, using multiple perceptual modalities to communicate their wishes easily and intuitively.

Pervasive knowledge *access* enables humans to access, anytime and anywhere, facts and other knowledge useful in personal and professional lives.

Machine-to-machine transactions automate mundane information functions, as well as control of our physical environment, to suit our wishes by following our scripted instructions.

Human-to-human collaboration helps people get together for discussions and decisions, track their interactions, manage the flow of information among them, and document their decisions and the rationale behind them across space and time.

5.1.2 Approaches used for the solution

Oxygen enables human-centred communication through a combination of perceptual interaction, individualized knowledge access, software agents and collaboration technologies. Speech and vision technologies enable us to communicate with Oxygen as if we're interacting with another person, saving much time and effort. Individualized knowledge access supports the natural ways people use to access information. In particular, it supports personalized, collaborative, and communal knowledge, 'triangulating' among these three sources of information to find the information people need. Software agents accept scripting commands and encapsulate objects, both physical and virtual, so that their actions can be automated. Collaboration technologies help us perform a wide variety of tasks that we want to do in the ways we like to do them by recording the context of human-to-human interactions.

Oxygen enables human-centred computation through merging person-centred devices, flexible networks and adaptable software systems, which dramatically extend our range by delivering user technologies to us at home, at work or on the go.

Person-centred devices include Computational Enviro21s (E21s) and Handheld (H21s) devices.

Computational devices create intelligent spaces inside offices, buildings, homes and vehicles. They provide large amounts of computation as well as interfaces to camera and microphone arrays, large area displays, and other devices.

Handheld devices provide mobile access points for users both within and without the intelligent spaces controlled by E21s. H21s accept speech and visual input, and they can reconfigure themselves to support multiple communication protocols or to perform a wide variety of useful functions (e.g., to serve as cellular phones, beepers, radios, televisions, geographical positioning systems, cameras, or personal digital assistants)

Flexible, decentralized networks (N21s) connect dynamically changing configurations of self-identifying mobile and stationary devices. Through algorithms, protocols, and middleware, they configure collaborative regions automatically, creating topologies and adapting them to mobility and change, provide automatic resource and location discovery, provide secure, authenticated, and private access to networked resources and adapt to changing network conditions, including congestion, wireless errors, latency variations, and heterogeneous traffic.

Adaptable software systems (O2S) intercept and adjust to changes in the environment or in user requirements. Software architecture provides mechanisms for building applications using composable, distributed components, customizing, adapting, and altering component behaviour, replacing components, at different degrees of granularity, in a consistent fashion, person-centric, rather than device-centric, security, and disconnected operation and nomadic code.

Speech and vision, rather than keyboards and mice, provide the main modes of interaction in Oxygen. Multimodal integration increases the effectiveness of these perceptual technologies, for example, by using vision to augment speech understanding by recognizing facial expressions, lip movement, and gaze. Computers recognize and classify features (e.g., faces and automobiles) and actions (e.g., gestures, gait, and collisions) in their field of vision. They detect patterns and object interactions (e.g., people entering a room or traffic accidents). For high-security transactions, where face recognition is not a reliable solution, a vision-based biometrics approach (e.g., fingerprint recognition) integrates sensors in handheld devices transparently with the Oxygen privacy and security environment to obtain cryptographic keys directly from biometrics measurements.

5.1.3 Results achieved

In order to transform the vision of human-centric computing into reality, the Oxygen team has been developing a set of prototype systems for matching the following technical challenges:

- Embedded, it integrates computing devices into the ordinary environment (e.g. wall-mounted and touch-sensitive displays with microphones, speakers, and cameras). In the demo of an intelligent room, people can speak with, gesture to, and interact with the 'invisible' computer. Smart sketching and design tools help extract design rationales from simple sketches and enable the room to record them.
- Handheld, a portable, Web-accessing, energy-aware and 'anonymous' physical object capturing device. A prototype H21, equipped with a microphone, speaker, camera, accelerometer, and display with perceptual interfaces, is built on the emerging hardware architectures of RAW (Raw Architecture Workstation) and Scale (Software-Controlled Architectures for Low Energy), StreamIt compiler and a resource discovery proxy INS (Intentional Naming System). The RAW and Scale expose hardware to compilers, which optimize the use of circuitry and power. StreamIt is a programming language and a compilation infrastructure facilitating the programming and compilation of large streaming applications. INS provides resource discovery based on what services do, rather than on where they are located.
- Adaptable, which provides flexibility and spontaneity, in response to changes in user requirements and operating conditions. Cricket is an indoor location system for pervasive and sensor-based computing environments, which provides fine-grained location information (e.g. space identifiers, position coordinates, and orientation). As we interact with devices or spaces, Cricket will adopt our information personalities and respect the space privacy.
- Privacy and security, which respect our desires for privacy and security. The SFS (Self-Certifying) and CFS (Cooperative File Systems) provide secure access to data over untrusted networks without requiring centralized control.
- Intentional, which enables people to name services and software objects by intent. The INS supports intentional message delivery (e.g. "Play that music on the nearest speaker"). Names in INS describe application intent in the form of properties and attributes of resources and data, not just network locations.

5.1.4 Assessment for Amigo

Oxygen is about reaching an ambient intelligent working environment through a combination of embedded, adaptable, energy-saving, intentional and security-aware system technologies. In contrast, Amigo focuses on making an ambient intelligent home real through bringing the usability of advance technologies and the attractiveness of intelligent services together. According to the results by Oxygen and the three scenarios by Amigo, the following features in Oxygen might be shared for facilitating Amigo:

- Embedded, perception interaction allows home people to move physical electronics into the background.
- Adaptable, INS provides home people to locate what services they need, rather than where they are. SFS and CFS provide home people secure, private, and efficient access to networked devices. Abstraction and specification-based Oxygen software architectures provide operational support for application changes and device customization.
- Attractiveness is the key drive for puting Amigo into the market. Smart knowledge access, INS and StreamIt technologies in Oxygen enhance common knowledge access, device service support and stream data management. Those might be configured or extended to improve Amigo attractiveness.

5.1.5 Challenges for Amigo

Oxygen is very different from Amigo in the aims and involved domains. Oxygen tries to provide a pervasive, human-centred computing environment and increase work efficiency. Oxygen emphasizes ubiquitous computing by making our ambient objects like PCs, whiteboards, office devices and meeting rooms intelligent.

Amigo envisions an easy, free and experience full life through networking and smarter home electronics. Amigo thinks of attractiveness as an important challenge. Besides meeting the requirements of embedded, perceptual interaction, adaptable software, and hardware mobility, Amigo also should take the following into consideration:

- Interoperability: Amigo needs to provide domain stakeholders unified and open standards for making, using, configuring, integrating and interacting home electronics. The standards must be accepted easily and willingly.
- Intelligent services: Amigo mainly focuses on developing intelligent services for easy life, which might be customized, followed and respects privacy.
- Semantic information, Amigo aims to bring home people ambient intelligence. More attention should be paid to the interaction at the semantic level between people and ambient intelligent devices.

5.2 Ambience

The ITEA AMBIENCE (Context Aware Environments for Ambient Services) project has the goal to jointly develop capabilities needed for the creation of integrated ambient intelligent environments. The project partners have generated concepts of such environments, and investigate architectural methods and tools that allow their future development. To validate the concepts, the required technologies are integrated into operational demonstration and evaluation systems. The consortium focuses on indoor home-, professional- and public-domain applications.

5.2.1 Concepts used

AMBIENCE aims at electronic systems and environments that are sensitive and responsive to the presence of people, i.e., are context aware, adaptive and intelligent (ambient intelligence).

Ambient intelligence is an exciting new concept in information technology, in which people are empowered through a digital environment that is aware of their presence and context. The environment is sensitive, adaptive and responsive to their needs, habits, gestures and emotions. The issues posed by ambient intelligence require multi-disciplinary and multi-cultural research, with input from computer science, electrical engineering, interaction design and behavioural studies.

The main issues tackled in Ambience are:

- **Ubiquitous systems with natural interaction**

Ambient intelligence merges two important trends: ‘ubiquitous computing’ and ‘social user interfaces’. It builds on advanced networking technologies that enable robust, ad-hoc networks to be formed by a broad range of mobile devices and other objects (ubiquitous/pervasive computing). By adding adaptive user-system interaction methods, based on new insights into the way people like to interact with computing devices (social user interfaces), better digital environments can be created. These context-aware systems combine ubiquitous information, communication, and entertainment with enhanced personalization, natural interaction and intelligence.

- **Architectures and methods for context-aware environments**

Ambient intelligent environments support ubiquity, awareness, intelligence, and natural interaction. Ubiquity means being surrounded by numerous interconnected embedded systems that are invisible (in the background). Awareness means that the system can locate and recognize objects, devices and people, and understand their intentions. Intelligence means the digital environment is able to analyze the context, adapt itself to users, learn from their behaviour, and eventually recognize and, perhaps, even show emotion. Natural interaction refers to providing functions, such as speech and gesture recognition, as well as speech synthesis.

5.2.2 Approach used for the solution

In Figure 5.1, the decomposition of an Ambient-Intelligent system is depicted. This decomposition has been created by analyzing the requirements for Ambient-Intelligence systems in the AMBIENCE project, and was validated by the Ambience demonstrators.

This decomposition serves as a reference model, identifying the key functions needed in ambient intelligence systems, and placing them in the layered reference model shown before. Note the orange-coloured bar on the right-hand side. This bar depicts functions that are not directly linked to a specific layer, because of their end-to-end nature.

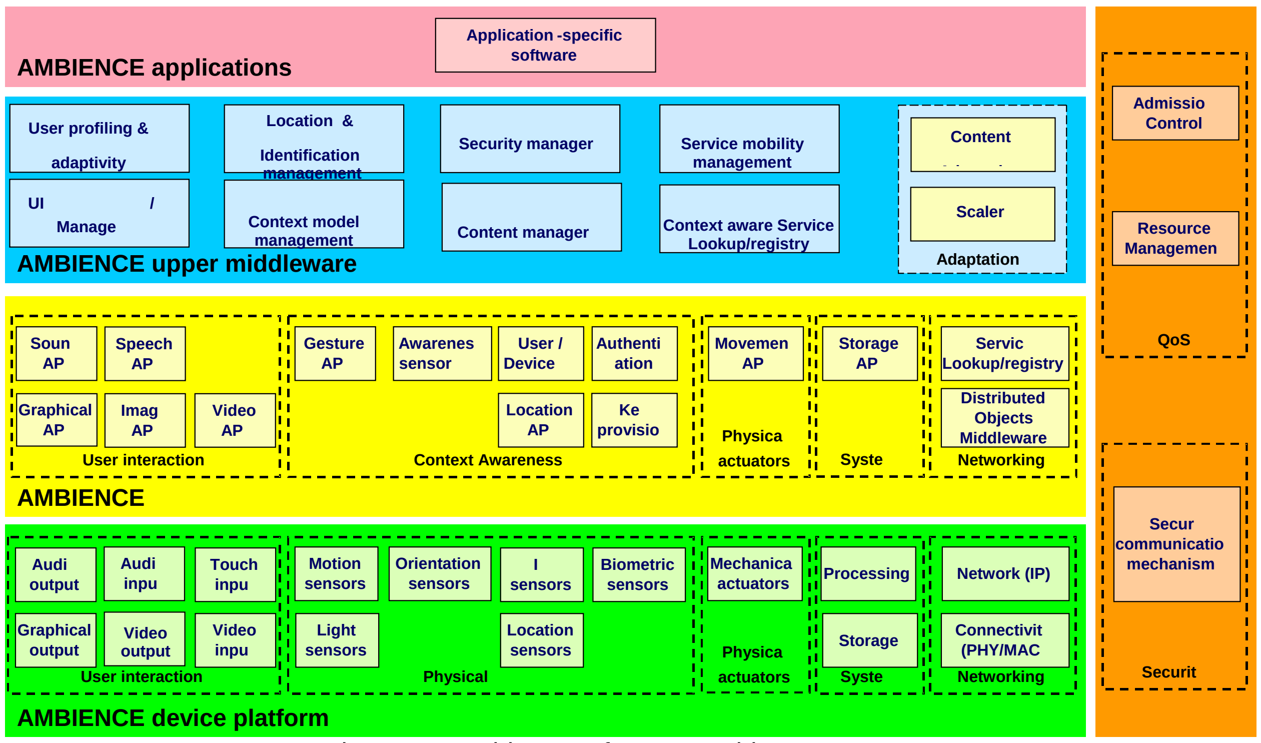

Figure 5.1 Ambience reference architecture

The distributed service infrastructure and the basic system services modules are shown in more detail in Figure 5.2. The distributed service infrastructure is the basic infrastructure needed to deal with services in a distributed system, whereas the basic system services are a collection of generic services that is useful, but not essential to distributed systems.

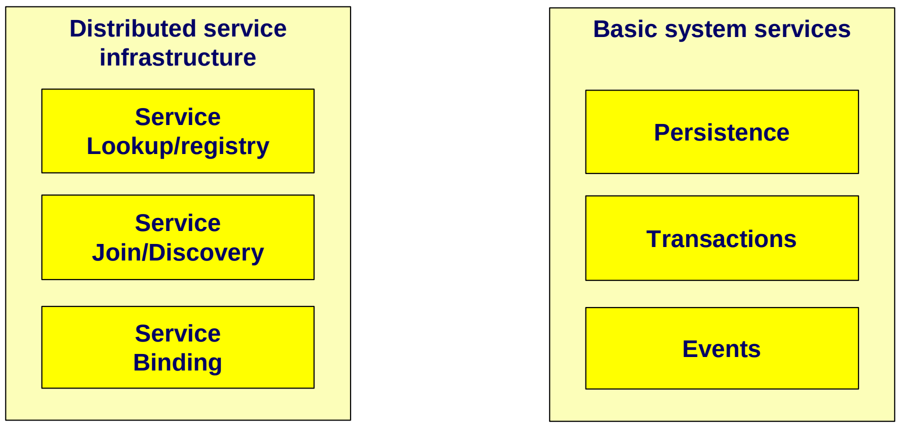

Figure 5. 2 Detailing of the distributed service infrastructure and basic system services

In Figure 5.3, the collaboration of the functional units of Figure 5.1 is shown, by means of arrows. The arrows indicate "calls" or "uses" relations. The colours of the blocks indicate the layer the blocks sit in. Note that, for reasons of simplicity, not all functional units in the Ambience device platform layer or the Ambience platform layer are included.

This is especially true for user-interface devices and awareness sensors, because there is a large variety of both of these, and the exact nature of a sensor does not inherently change the collaboration at higher levels.

Note that the collaboration depicted here is specified at the conceptual level, leaving the actual implementation, deployment structure (e.g. ranging from fully centralized to completely distributed), and mechanisms (e.g. synchronous method calls or asynchronous message passing) unspecified.

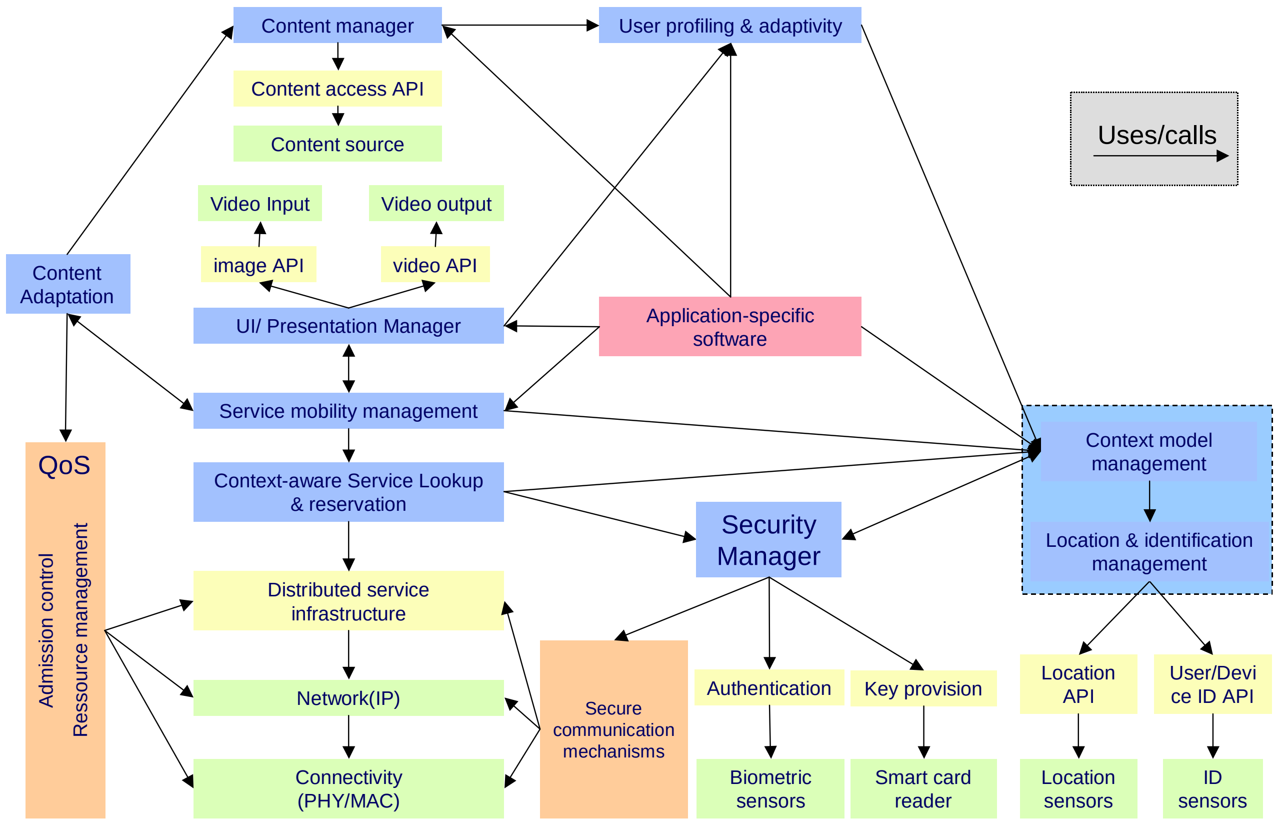

Figure 5.3 Collaboration of functional units

5.2.3 Results achieved

The AMBIENCE project jointly created networked Context Aware Environments. It generated concepts and developed architectures, methods and tools. To validate the concepts the required technologies were integrated into operational systems, and were demonstrated on systems for home, office and public building environments.

Middleware architecture

The main structure of the architecture is a layering into four layers, of which the lower three can be regarded as the ambient intelligence platform, and the fourth (top) layer consists of components and services that are more or less application-specific. Two topics, quality of service and security, defied the layering, and were organized in a vertical structure, due to their end-to-end nature.

The architecturally most innovative layer is clearly the Ambience upper middleware layer. This is the layer that integrates many technologies and infrastructures, adds intelligence and learning, and ultimately forms the platform on which ambient intelligent applications and services can be built.

The two key aspects of ambient intelligence, natural interaction and ubiquity can be clearly recognized in the various components: user profiling and adaptivity, the UI/Presentation manager, the location and identification manager, and context model management deal mainly with natural interaction, whereas the security manager, the content manager, service mobility management, the context-aware service lookup/registry and the adaptation blocks deal mainly (but not exclusively) with ubiquity.

With respect to the lower layers, the innovation is much more in specific sensor, imaging, and network technologies, and not so much in the architecture.

Demonstration of challenging results

Challenging results were achieved in the area of ubiquity, context awareness, intelligence and natural interaction. The project achievements were demonstrated in two demonstrators in each operating area, such as the mobile, professional and the home domains.

The two **mobile domain** demonstrators were called “Guide to a Meeting” and “Indoor Navigation”. The first demonstrator was created at the Philips Research Lab in Eindhoven, the Netherlands. A location-aware conference delegate support system, it was comprised of a robust and modular integrated server that used an architecture, inspired from the Web Services model, biometrics access control, wireless connectivity using mobile robot routers to optimise Quality of Service (QoS) and a ZigBee-based Radio Frequency (RF) localisation system. The second demonstrator, developed at the France telecom R&D site in Grenoble, integrates a location-technology independent location-management system together with a Personal Digital Assistant (PDA) based indoor navigation application that includes a Scalable Vector Graphics (SVG) player for the interactive scalable display of visual navigation data.

The two demonstrators for the **professional domain** were called “Intelligent Meeting Room” and “Smart Design Studio”. The first acts as a joint demonstrator. It was developed at the Barco site and connected to a remote office of KU Leuven using a robot (MakTub) mediated link. The other demonstrator was developed at the design studio of Italdesign-Giugiaro and demonstrated a highly interactive design approach for cars through the use of a broad range of interaction modes, including speech, gesture, tangible objects and a dedicated digital pen for the wall-sized display used.

Two demonstrators were also created for the **home domain***.* These were named “Ambient Intelligent Home” and “Multimedia Browser”. The first one was developed at the Philips' HomeLab in Eindhoven, the Netherlands and includes the robotic assistant "Lino". A range of entertainment, communication and personal health applications have been demonstrated, and evaluated in part. The second demonstrator was developed at the Thomson site in Rennes, France. In this, a speech recognition module and a virtual presenter were integrated successfully in a ‘movie-recommender’ interface, which, as well as content navigation and recommendation modules, used textual feature extraction and vision-based user recognition.

5.2.4 Assessment for Amigo

The Ambience reference architecture can be considered as a first step towards the more advanced ambient intelligence architecture targeted in Amigo. The reference architecture pictured above did identify all relevant building blocks, but did not enforce strict consistency across all demonstrators, and as such is more a framework than proper architecture. Each of the demonstrators included a distinct subset of the building blocks and made different choices for the implementation of these blocks. Some of the demonstrators did not include a full-fledged service advertisement and discovery infrastructure, and so they could not be configured in a fully dynamic fashion. Also the variety of physical sensors and actuators used was limited. Context-awareness has been limited to personal context (profiles, preferences, interaction history) and location and did not yet integrate the management of all pieces of context information in a consistent model. In some of the demonstrators (e.g. guide to a meeting), location has been managed in an ad-hoc vertically integrated fashion (applications interacting directly with location-sensing technologies). In some others (e.g. the indoor navigation demo) it was managed in a more general way with a prototype location management infrastructure that mediated between location-sensing technologies and applications, making it easier to generalize the use of location-adaptiveness across various technologies and applications.

A similar comment can be made regarding the demonstrators and scenarios proposed by Ambience, both of which were proposed at the beginning of the project and those that were implemented at the end of the project. They are a first step towards implementation of the ambient intelligence concepts but fall short of fulfilling their more far-reaching ambitions.

Most of the Ambience scenarios and demos can be characterized as being still device-centric, in as much as they enforce coupling between an application and a device, or constrain user interaction to a single device or a limited, pre-configured set of devices.

5.2.5 Challenges for Amigo

- Enforce a single consistent architecture across all three demonstrator domains.
- Enforce the use of a full-fledged service-oriented paradigm to ensure dynamic configurability of devices and services.
- Propose a general model for context management that makes context information usable at different levels of abstractions across the full spectrum of context-data acquisition technologies on the one hand and context-aware applications on the other hand.
- Fulfil the smart space vision of distributed interface devices, relaxing the device-application coupling.

5.3 Ozone

The objective of the IST FP5 Ozone project was to support the effective use and acceptance of ambient intelligence in the consumer domain. Towards that goal, the Ozone project defined and implemented a framework to enable consumer-oriented ambient intelligence applications. The Ozone framework consists of three architectural layers that were designed and (partly) implemented in the project:

- The top layer takes care of service enabling with an emphasis on context awareness or sensitivity.
- The middle layer is responsible for the software environment where seamless task migration is a crucial issue.
- The bottom layer delivers a powerful computing platform where high performance computing at a low power level is the differentiating factor.

5.3.1 Concepts used

Concepts implemented in the framework focused especially on:

- **User-centric** retrieval and consumption of information compared to the current practice: a computer-centric approach.
- **Natural interfaces** that put the user in the foreground and the system in the background.
- Terminal and network **QoS**.
- **Pervasive service** provisioning enabling the deployment of and access to services in the mobile situation.
- Powerful, but **energy-efficient**, computing.

5.3.2 Approach used for the solution

The Ozone project aimed to develop solutions towards a first-generation consumer-oriented ambient intelligence system. The ambient intelligence vision itself ultimately implies new, intelligent environments that are sensitive and responsive to the presence of people and to the context of use. The environments enable natural interaction facilities for people, and in this way realize an infrastructure that seamlessly and ubiquitously aids users in their daily activities. The Ozone device architecture was already based on the Ambience results and gave a good impression of all functionalities that could be present in such a device (see Figure 5.4).

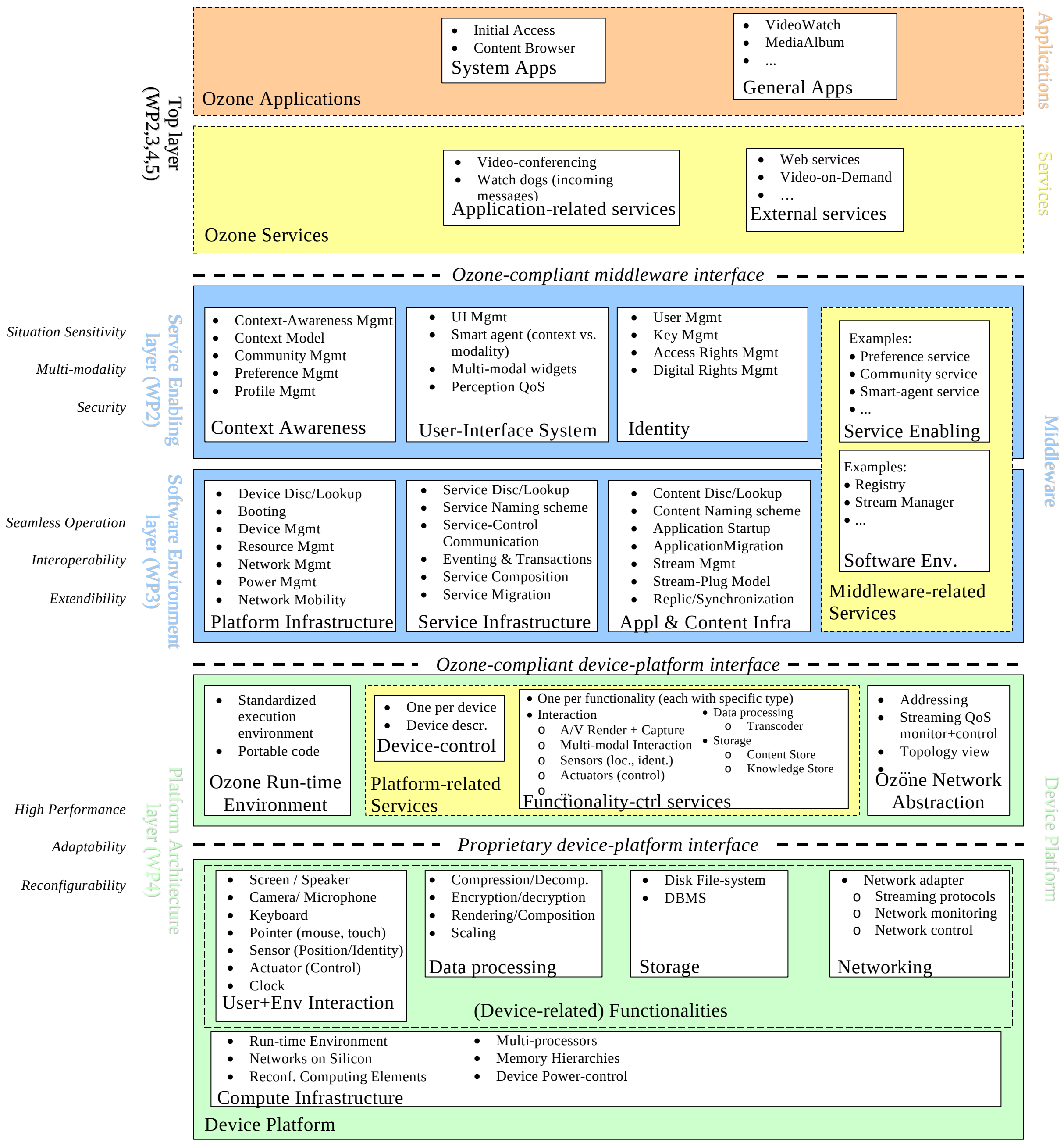

Figure 5.4 Ozone device architecture – a layered functional decomposition.

Interoperability within the Ozone project is addressed by assuming that networked devices are all Ozone-compliant, i.e., run the Ozone distributed middleware enabled by layer 2 of the architecture (see Figure 5.5). In order to support pervasive service provisioning, the Ozone middleware is based on the pervasive Web services architecture, i.e., Ozone services are compliant with the Web services architecture, which is extended to cope with the specifics of service provisioning in the mobile wireless environment. An extensive QoS architecture was further defined and implemented to support the exchange of and access to multimedia content (see Figure 5.6), which was considered as a key requirement for consumer-oriented ambient intelligence. The Ozone demonstrators in particular showed several forms of both network and terminal QoS of use for the in-home closed network and prepared for the home broadband open access network.

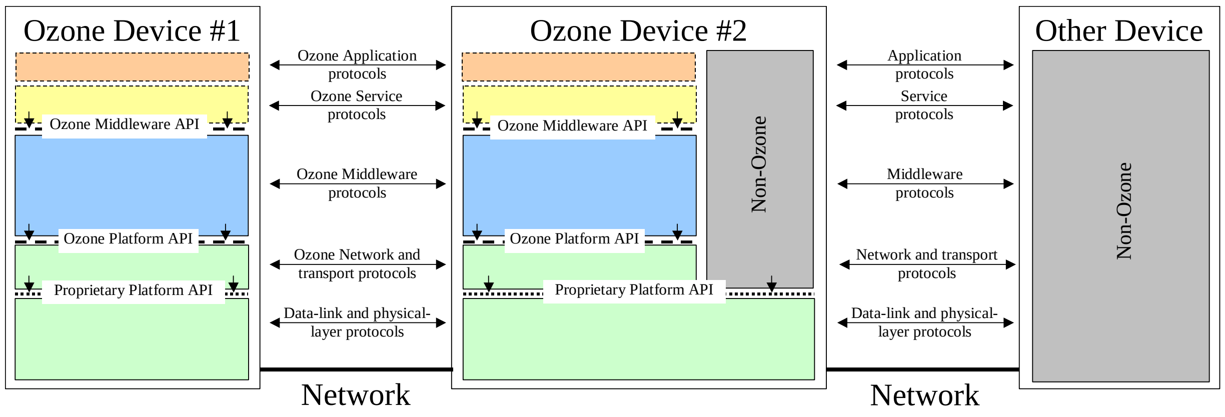

Figure 5.5 Network of Ozone and non-Ozone devices.

Adhering to the same protocols at all levels ensures interoperability between devices across the network. To enable re-use of layer-implementations by higher layers, APIs can be defined. These APIs may expose both the lower-layer protocols of the network connection and specifics of the underlying device platform.

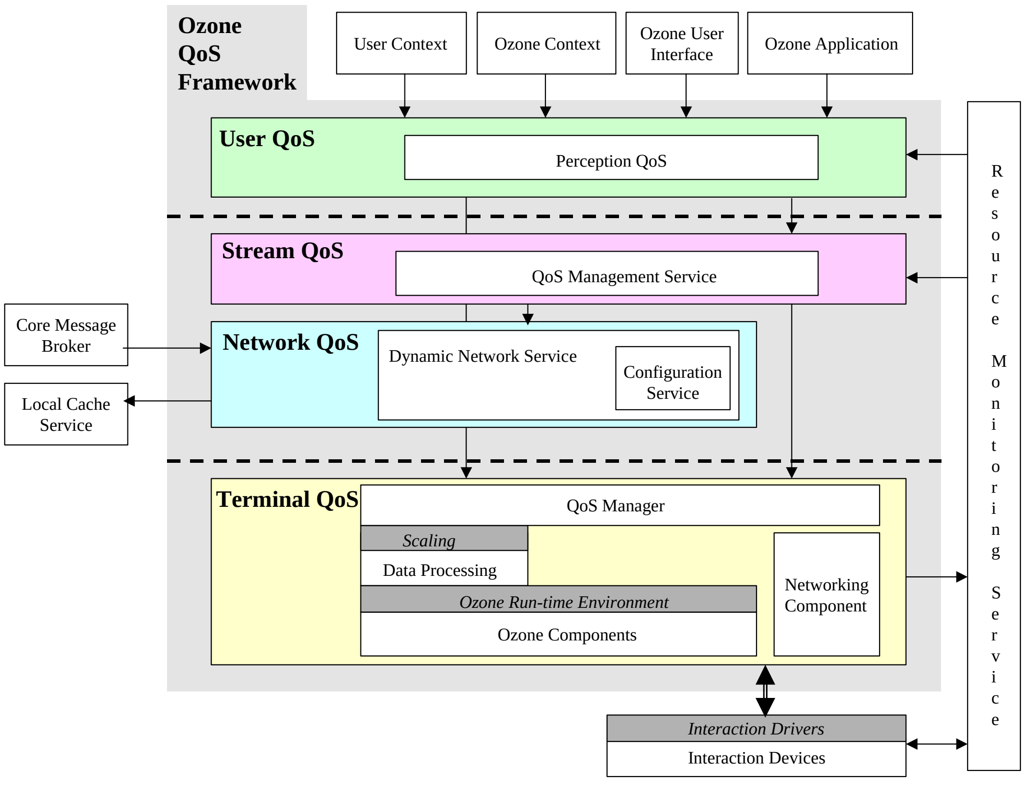

Figure 5. 6 QoS view of the Ozone architecture.

5.3.3 Results achieved

The main results achieved by the Ozone project are:

- Advances in content adaptation and interface technology, enabling content to be adapted to user preferences.
- Enabling users to control all the content and devices without technical knowledge.
- Enabling interfaces that are multi-modal, intelligent (learning from behaviour) and interactive.
- Improved access to services, enabling access from and deployment on new portable, light, friendly devices, via Internet portals in different places.
- Making personal content, professional work, entertainment data and news information accessible while on the move.
- Providing new user experience, as content is presented differently depending on the device, where the user is, and on the time of the day.
- A new distributed framework proposal that is able to manage networks, content, services, and devices.
- New device architecture that extends the 'one content to one device' paradigm to 'one content to many devices'.

In general, the Ozone project was a good step forward by providing a distributed framework that is able to manage resources in networks, devices and content and services. With supported intelligent interfaces and context awareness, huge improvements have been delivered. However, not all results have been integrated together, in particular lacking integration of pervasive service provisioning with QoS provisioning, and of context management with pervasive service provisioning.

5.3.4 Assessment for Amigo

The results of the Ozone project are used as the starting point for the Amigo project. As Ozone already took the Ambience architecture results into account, it can be seen as a second step towards Ambient intelligence. Within Amigo however the focus will be much more on the middleware and services, while in Ozone, the major block of work has been done in the platform layer. This layer can now be re-used for a large part within the Amigo project.

The Ozone middleware was designed so as to allow service access by the nomadic user in most situations. This has led to the development of the Ozone WSAMI[5] middleware, which supports the abstract specification of ambient intelligence applications in the form of software architectures, together with their dynamic composition according to the environment. The WSAMI middleware builds on the Web services architecture, whose pervasiveness enables service availability in most environments. In addition, the dynamic composition of applications is dealt with in a way that enforces quality of service for deployed applications in terms of security and performance through the systematic customization of connectors that dynamically integrate relevant middleware-related services. Although the WSAMI middleware is service-oriented and builds on the pervasiveness of the Web, it is too restrictive to address the requirements of the Amigo system, which should integrate software services from the CE, domotic, mobile and PC domains.

In general, the Ozone project focussed primarily on the CE domain, while the Amigo project takes the CE, PC, mobile and domotic domains into account, which requires addressing interoperability among related software services at the middleware layer. Intelligent user services have furthermore not really been in mind.

It is moreover important to realise that the middleware was not enforced strictly over all demonstrators and was meant to be a framework, as opposed to one consistent middleware as Amigo has as one of its main objectives. In particular, the issues of pervasive service access and QoS management relevant to the middleware layer were investigated independently and not integrated.

[5] WSAMI - Web Services for AMbient Intelligence. Available at http://www-rocq.inria.fr/arles/download/ozone/index.html

The results of the Ozone project have laid groundwork for the Amigo project objectives: for example, the impact of security and privacy for these kinds of systems has initially been underestimated. The limited work done within the Ozone project on security and privacy was only able to provide a basic architecture for this. Within the Amigo project, delivering an architecture that is truly secure and private in nature, as it has been taken into account from the beginning is one of the goals. During the Ozone project, we also became aware that the multi-user environment the average home is, has a serious impact on the implementation of a networked system. Multiple users should be able to access services, content etc. at the same time. In an environment of limited resources, e.g., wireless network bandwidth, they should not hinder each other in unacceptable ways. Also, heterogeneity in the home network that integrates devices from various domains will remain. It is thus crucial to devise a middleware solution that allows integrating heterogeneous software services, as opposed to introducing a middleware to be deployed on all the networked nodes for them to be actually networked within the system.

5.3.5 Challenges for Amigo

Various challenges remain for the Amigo system, when taking the Ozone framework as a starting point. These include:

- To design and implement (open source) interoperable middleware that ensures interoperability over all four domains: CE, PC, mobile and domotic
- To ensure dynamic plug and play of devices, networks and services.
- To ensure a multi-user system
- To ensure a proper security and privacy architecture and implementation
- To offer effective intelligent user services, in particular dealing with context-awareness.

6 Conclusion

Amigo aims at merging currently separated home appliance domains and advanced information technologies to offer home users an ambient intelligent environment for realizing a modern excellent daily life (e.g. ubiquitous services, natural interactions and personalized computing). In order to reach this objective, this report presented the state-of-the-art of relevant background technologies and building blocks consisting of:

- Stationary and wireless computing nodes (e.g. PDAs, PCs, TVs, game consoles, smartcards, sensors, and actuators)
- Networking technologies (e.g. home networks, IP-based protocols, real-time protocols, ad-hoc routing and hybrid protocols, mobility support)
- Operating systems (e.g. Linux, Windows, Symbian)
- Middleware service oriented architectures (e.g. OSGi, UPnP, Web Services including service composition and semantic service modelling)
- Service discovery protocols (e.g. SLP, Jini and SSDP)
- Support for security and privacy, QoS, content management, and accounting and billing
- Intelligent user services (e.g. context management including semantic context modelling, multimodal user interfaces, user modelling and profiling)
- Software architectures aimed at ambience intelligence (e.g. Oxygen, Ambience, Ozone)

Also the assessment and challenges for further Amigo research and development are identified. To summarise:

- Most of the computing nodes, operation systems and networking technologies will be probably reused in the developed Amigo system. The main goal in this technological area for Amigo is to identify an appropriate technological solution for the particular scenarios and applications (e.g. suitable wireless access technology, infrastructure/infrastructureless operation mode, particular routing algorithms, network configuration protocols, specific groups of computing nodes and sensors)
- The architecture for the Amigo system will likely be based on services oriented architectures including Web Services, UPnP and OSGi, reusing existing service discovery protocols such as SLP, Jini and SSDP. One goal for Amigo is to allow the interoperability of services and applications based on different protocols. Amigo devices should be able to interact with heterogeneous devices and heterogeneous networks in the domains of domestic appliances and consumer electronics. Amigo devices should in particular be interoperable with consumer electronic devices following DLNA recommendations, which means UPnP devices and control points. Interoperability between UPnP, Jini and Web Services is still an open issue. Full interoperability may require semantic matching between varied services descriptions
- Security & Privacy plays an important role in the acceptance of an Amigo system as the next Ambient Intelligent system. A middleware architecture dealing with heterogeneous networks and handling privacy sensitive information like context and user profiles needs to prove it can handle this information in a secure way respecting the users privacy. A multitude of excellent security & privacy mechanisms and technologies exists as of today with their own drawbacks like high performance requirements, centralized authorities, key management or complex maintenance. In Amigo, a security and privacy architecture has to be designed as secure as the existing ones but suitable for a networked home environment.
- QoS management is a necessity for every environment, including the home. It concerns the ability to obtain service-level guarantees such as timeliness, availability, fault-tolerance, and survivability for applications executing in such environments. One of the challenges for Amigo is to enforce QoS in terms of at least security and performance regarding resource consumption, i.e., the two mandatory criteria for the consumer acceptance of ambient intelligence systems

- The different aspects of traditional content management (creation, manipulation, storage and distribution) are all applicable in a home scenario. The challenge for Amigo in relation to content management is to design and implement a process workflow specific for the home scenario. Limited or no research has been done up to today in this area since it was usually driven by business requirements. For Amigo, the user scenarios and derived requirements need to be carefully analyzed in the light of usability, performance, privacy, security and QoS
- Context management is a key enabling technology for Amigo. Intelligent services and applications need support for this, and therefore a generic context management framework is a required feature of the Amigo middleware. It is a key challenge for Amigo to design and implement a context-management infrastructure that can cope with several types of context information, and provide for the secure exchange of this information while protecting the privacy of end-users
- One of the key characteristics of the ambient environment targeted by Amigo is natural interaction. The naturalness of user interactions shall clearly be improved, as nowadays technologies do not support adequate features to capture and interpret natural communications between and with humans. This includes the development of efficient techniques to support multimodal speech and 2D and 3D gesture recognition, and to improve the overall processing chain from speech acquisition to natural language understanding. Efforts have to be realized to support and enhance the emergent paradigm of implicit and invisible interactions, which implies (semi-) automatically building context models by tracking, observing and interpreting the user's behaviour. This is actually much more difficult to achieve than traditional direct interactions, where the user is consciously talking to a computer system, and makes cognitive efforts to clarify their inputs
- To make the Amigo system more friendly and attractive for the end-user, the concept of user personalization needs to be addressed. In order to provide personalization and corresponding personalized services, it is necessary to combine context information with information about users via user modelling and profiling services. Profile information has to be classified and corresponding methods for managing and accessing profile data have to be developed
- Service provisioning as enabled by the Amigo middleware is required to deal with the mobility of users, which creates specific challenges to be addressed such as the management of service mobility/handover and location management, where an appropriate location detection technology should be selected based on the application and environment
- The reference architectures and results achieved in the Oxygen, Ambience, and Ozone projects can be seen as a first step towards the more advanced ambient intelligent architecture targeted in Amigo. These projects focused primary on the CE domain, while Amigo takes the CE, PC, mobile and domotic domains into account, which requires addressing interoperability among related software services at the middleware layer taking into account also security and privacy requirements, QoS provisioning and to offer effective intelligent user services, in particular dealing with context management that makes context information usable at different levels of abstractions across the full spectrum of context-data acquisition technologies on the one hand and context-aware applications on the other hand.

Amigo systems are full of ambition to offer a class of information pervasive and experience-sharing home applications to householders by providing interoperability, self-administration, extensible, and high QoS services with a natural user interface. Enjoying music according to tastes, mood and context but regardless of the source, across different devices and locations in the home is an obvious example. These make Amigo technical complex and multidisciplinary in knowledge.