IST Amigo Project

Deliverable D3.1a
Detailed Design of the Amigo
Middleware Core
Service Modelling for Composability

Public

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

Deliverable Number	:	D3.1a
Title of Deliverable	:	Detailed Design of the Amigo Middleware Core Service Modelling for Composability
Nature of Deliverable	:	Public
Internal Document Number	:	Amigo_D3_1a_v1.0.doc
Contractual Delivery Date	:	31 August 2005
Actual Delivery Date	:	13 September 2005
Contributing WPs	:	WP3
Author(s)	:	VTT: Jarmo Kalaoja, Julia Kantorovitch, Marko Karjalainen TID: José María Miranda, Álvaro Ramos NTUA**:** Miltiades Anagnostou, Ioannis Papaioannou, Ioanna Roussaki, Dimitris Tsesmetzis Ikerlan: Jorge Parra FT: Fano Ramparany

Abstract: The objective of this document is to start the analysis on vocabularies required in various Amigo domains (domotic, PC, mobile, CE) and also to model non-functional properties of the service to cope with the heterogeneity of services descriptions and requirements specifications. The vocabulary ontologies for the semantic modeling of Amigo services are aimed to complement the language for service specification discussed in deliverable D3.1b.

Keyword list: service modelling, ontology, modelling and reasoning tools, service composition.

Table of Contents

1 Introduction

Information appliances and broadband networks have become increasingly popular. Also home devices follow similar trends towards the stage, where they can constantly be connected via networks. The range of these devices is spanned from the simple devices that are mainly used to host the services and traditionally belong to the home automation (i.e. domotic) and consumer electronics (CE) domains, such as smart sensors and smarter appliances like heater, microwave oven, washing machine, fridge (i.e. domotic), and HDTV, game console (i.e. CE) to the devices which apart from providing many different kinds of services also can be used to control devices such as PDA, Smart Phone, Notebook, and PC that traditionally belong to the mobile and personal computing (PC) domains. Along with the services offered by devices (i.e. hardware) and domain specific software services, in the networked home there are a numerous kinds of other services so called intelligent user services, such as context management, awareness and notification, user modelling and profiling, etc, which aimed to bring a new kind of interaction supported by ambient computing technologies into our homes thus enabling domestic users to enjoy their life, providing higher level of comfort and convenience.

As stated, the main goal of Amigo project is to merge the traditionally separated domains of home automation, consumer electronics, mobile communications, and personal computing, to offer home individual residents user-friendly, intelligent, and meaningful interfaces to handle home information and services. To achieve this goal, the heterogeneous networked services in the home, where is a wide variety of heterogeneous networks, devices and software infrastructures, should be discovered, invoked, integrated and composed in a ubiquitous and seamless manner to address the user requirements. The semantic modelling of Amigo services enables efficient automated execution of the aforementioned tasks. Our solution exploits the semantic capabilities offered by the Web Ontology Language (OWL) [OWL] to effectively describe services and resources located in home environments. Modelling services using ontology based approach allows to semantically describe services so that their semantics are independent of service implementation details, programming languages, underlying operating systems or middleware infrastructures. Specifically, OWL-based ontologies that model the functional behaviour of Amigo services through the modelling of components and connectors have been introduced in [Amigo-D2.1]. Components abstract services and connectors abstract interaction protocols. As in traditional software architectures, service modelling is performed by the operations that it provides and requires from the environment, the Amigo components (i.e. services) have been further modelled based on the number of required and provided capabilities. Each capability specifies a number of inputs and outputs. The capabilities along with their inputs and outputs can be used for the service discovery process (i.e. capabilities/requirements matching or conformance verification). As was underlined in [Amigo-D2.1], the non-functional properties of the service, namely quality of service (QoS), context in which a service is executed, and preferences specified in user profiles are essential requirements in Amigo. The QoS characteristics of the service related to its performance, configuration, cost and security features should be used for dynamically selecting the service that best meets the user needs. The context information has to be taken into account during the matching/selection process even if the original request doesn't explicitly specify the context of usage. Location is a classical example of context information which has usually an impact on the selection of a particular service. Other types of context information that are important and can be considered in the home environment are time, temperature, weather, and user activity, sometimes referred as "current user status", number of people in particular places, etc. Finally, the preferences and interests of domestic users should be taken into account in service provision through the aggregation of their profiles into the service discovery process. Integration of non-functional parameters into the ontology languages designed for the Amigo services is currently being studied. This issue is further analysed in deliverable document D3.1b [Amigo-D31b].

In order to allow a rich representation of services and thus facilitate efficient service discovery and composition, functional capabilities, inputs, outputs and non-functional attributes of the services can be further semantically annotated using external vocabulary ontologies. The use of ontologies enables computational entities and services to have a common set of concepts and vocabularies for representing knowledge about a domain of interest, while being able to interact with each other. By using such ontologies, the relationships between entities can be more clearly expressed and these allow for better reasoning on their properties. Based on ontologies, higher level knowledge (i.e. situation) in the specific domain can be deduced by interpreting and reasoning on the domain concepts. Ontologies are also beneficial for the re-usage of knowledge, as several ontologies from various sources can be integrated to describe the specific domain. Finally, the deployment and customization of the Amigo system in any home can be considerably facilitated by using a common set of concepts and vocabularies developed for the networked home environment. Of course, ontology-based service modelling and discovery requires ontologies to be complete and carefully validated against application requirements. This document elaborates on the vocabulary ontologies for the semantic modelling of Amigo services. These vocabularies are aimed to complement the ontology language for service specification discussed in deliverable D3.1b [Amigo-D3.1b]. The objective of this document is not necessarily to deliver a set of new ontologies and classifications, but also to integrate and extend the existing ones, and provide some recommendations for service developers on how to use them.

Moreover, it is important to highlight that software architectures should be developed considering the concerns of the various stakeholders associated with the software system. For Amigo, some examples of these stakeholders are Amigo users, the developers of software for the Amigo aware devices, the developers of the Amigo intelligent services, and the developers of the Amigo applications. Amigo users will be confident with the Amigo system, as it is very easy for them to understand the rationale behind its behaviour. For the developers appropriate documentation needs to be prepared that is understandable and hides details irrelevant of the particular development task. The availability of the development tools which are robust, simple and easy to use is another critical issue. These give the opportunity for SMEs, research communities, and some individuals to create and deploy new services and applications for our daily home life. The open source tools for software semantic modelling and visualisation will be used to provide various stakeholders the viewpoints on Amigo system that conform their particulars concerns.

1.1 Document structure

Work on developing a set of ontologies, modelling concepts and specific domains, which can be used as a general-purpose vocabulary for describing Amigo services, is in progress. Relevant initial results, strategies, methodologies, and future plans are presented in this document.

In the following, the analysis on vocabularies and classifications for the domain specific services (i.e. domotic, CE, PC, mobile) and semantic modelling of non-functional properties of the Amigo services such as QoS and context information are discussed in Chapter 2.

Chapter 3 is dedicated to methodology for semantic modelling of services. The existing methods and tools, including their shortcomings and benefits are overviewed.

Chapter 4 concludes the document providing an overview of the current contribution and the work progress. It also presents the future plans for semantic ontology based modelling of functional and non-functional properties of the Amigo services, towards the dynamic ‘ad hoc’ service composition, adaptive to information related to user and services. Some approaches for adaptive service composition are outlined. However, as the research in this direction is in its early stage, the candidate solutions will be further studied and assessed.

2 Semantic modelling of Amigo services

Amigo scenarios [Amigo-D1.1] contain several scenes that illuminate how the Amigo system should interact with people in a service-rich home environment. Based on these scenarios, the Amigo system can be visualized not as a monolithic system, but as a system consisting of a number of separate specialized "assistants" or "agents", which demonstrate various properties and functionalities in support of their specific roles.

The Amigo system (acting as Entertainment Assistant) can start a day with playing music from preferred play list, favourite radio host, showing the personalized news or general summaries of hot news topics. It can configure the program so that it follows the user everywhere in the house or, even in order to follow him/her in a mobile or portable device outside the home area. It keeps track of play list, scores, and the persons the user has been playing with or providing usage instructions to newly obtained/subscribed games. It downloads personal profiles and integrates game devices when friends are coming over to the user’s house for interactive multi-player game sessions. It helps to select games to play and supports the usage of various displays in the home environment during the game session; so that players get an audio-visual feeling of the movements that are being made. It can set-up video-conference sessions for people to watch TV together and share the newest acquisitions of their collections, or just communicate with each-other.

The Amigo system (acting as Household Assistant) can select the correct settings for the session duration, power supply, the input material dosage, and the temperature for home appliances. It may even know how to detect the presence of inappropriate objects inside or near the appliances. It downloads recipes and cooking programs to the kitchen and displays them to facilitate the food preparation procedure, e.g., supporting users to cook while having video assistance. Moreover, the recipes always take the status of the provisions in the kitchen into account. It can maintain the overview of the food and household stock and generates shopping lists at predetermined time intervals. The shopping lists are personalized, while they also consider special offers, seasonal variations and nutritional balance requirements.

The Amigo system (acting as Ambience Assistant) can configure the home environment during game sessions, or while the user is watching TV by adapting the light, sound and video features/levels throughout the room. It can adapt the home environment and create an all-surround audio, light and video experience. It can also support content (music, film, TV/radio program or game) following a person everywhere in the house.

These roles should not depend on the availability of any specific set or types of devices. In order to support this, we introduced the "Engineer" actor in the Amigo system that acts as a supporting role for higher level Amigo applications and hides the necessary underlying technologies. Examples of what this actor may handle include the following:

- Control specific appliances such as blenders, lights and washing machines at a desired time.
- Set parameters such as duration, detergent dosage, and temperature for a washing machine session.
- Download a game to Amigo system in a PDA device.
- Download the music or program content on Maria's personal device.
- Control various lights as needed.
- Collect information from various detectors that are available.
- Etc, depending on domain.

This "Engineer" role is not a single application, it is rather handled by the Amigo Intelligent services, and a number of other services that are dynamically registered using the service discovery functionality of various appliances located in the Amigo house and composed by the Amigo system when necessary.

The objective of this chapter is to start the analysis of what kind of vocabularies are necessary in various Amigo domains, in order to cope with the heterogeneity of services descriptions (i.e. capabilities, inputs, outputs) and address the requirement specifications.

The example scene below illustrates the use of such vocabularies:

"John and Robert start to play an interactive multi-player adventure game in John's living room. The Amigo system adjusts the ambience for the game experience by controlling the lights and sound. Robert notices that it is late so he has to leave, but he continues to play the game during his metro trip."

Some examples of the vocabularies related to the various Amigo domains in this scene are:

- TV monitor, when plugged in, provides CE domain capability of *DigitalMediaRenderer* for *AVStream* for Amigo System.
- From context information and/or gesture recognition output Amigo UIs detect that John and Robert want to play. Upon activation of Amigo applications in the game console the system queries the capabilities of the Amigo Intelligent UI service considering the *Location* context provided by the Amigo User interface.
- When controlling the ambience in a room the Amigo system queries services providing the Domotic domain *ControlLight* and *LightDetectors* capabilities available in the context of that room.
- If it is day and there is too much light in a room, the ambience system queries availability of *ControlBlinders* Domotic domain capabilities in the context of that room and day time.
- To be able to support the continuation of the gaming session, John's Amigo system checks the *DeviceContext* of Roberts PDA. Amigo matches a most suitable mobile client technology provided by game console and uses the PDA's *ContentDownload* capability to load a game client and download status information to it. In order to support this, Robert's PDA needs to register to John's Amigo system, where the *QuestAccess* to services functionality of the security domain is being used to verify that the necessary access rights are in place.

A number of heterogeneous technologies as well as services and devices from different domains are involved in this short scene extracted from the Amigo scenarios. In addition, other services which are not that visible in the example above are seamlessly involved. For example, context management service will be used to provide user *Location* to allow user to continue his/her game in any place in the house or it will provide day *Time* in order to control the light in the room to enhance the user service experience. The Amigo system may ask the User modelling and profiling service for relevant user context and preferences about favorite adventure game to be automatically downloaded (i.e. *GetUserProfile* capability).

Different levels of ontologies (see Figure 1) can be envisaged to model the entire Amigo home environment appropriately in order to address the service/application developer's and user's requirements for efficient service discovery, composition and invocation. To verify and validate this approach, the following steps have been used in order to identify the required vocabularies and classifications on which the ontologies will be based [TUT]:

Step 1: Gather Amigo requirements for the various domain ontologies.

- Identify some domain-specific examples, based on Amigo scenarios, that could help to determine what types of information are required and what types of query support the information classification or ontology should provide.

Step 2: Consider available sources of classifications.

- List sources of information that might provide classifications, taxonomies or ontologies in the domain. Provide references to the information sources. Discuss available classifications from the Amigo point of view.

Step 3: Enumerate/list important terms in the ontology

- First create draft lists of the information based on various sources.

Step 4: Analysis and classification

- Analyse the lists you got from step 3 against the requirements from step 1 in order to identify the information needed by the selected example.
- Filter the list from step 3 according to the examples.
- Specify the kinds of ontologies (and relations between them) that are necessary for describing the information in this domain.
- Define a preliminary taxonomy or class hierarchy for the ontology.
- Try to identify functional and non-functional (QoS, context) aspects in the class hierarchies.

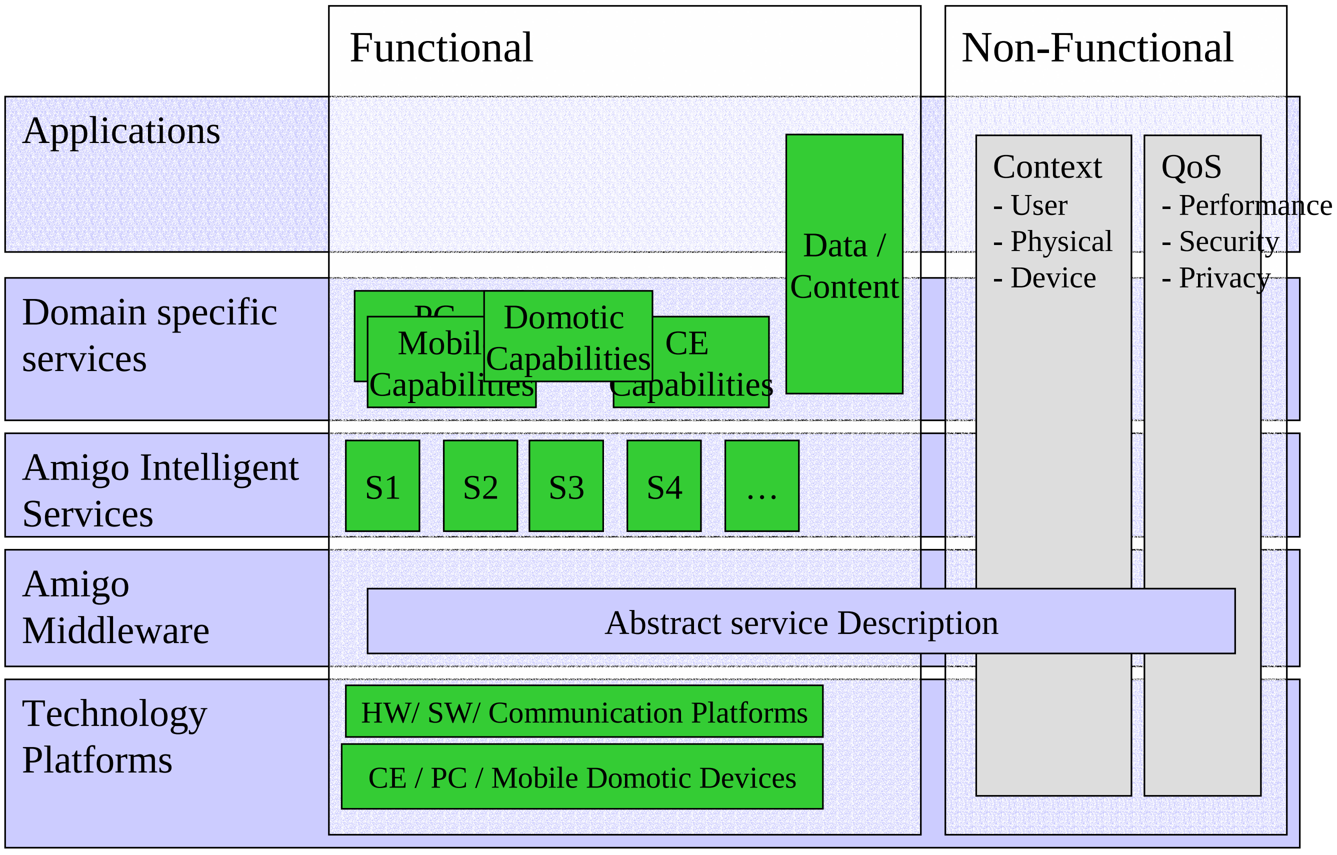

Figure 1 Different levels of services and ontologies

These steps are done iteratively. Thus, initially, we identify the devices and their functional properties and then we consider the context, QoS and non-functional capabilities in the domain. Information about the non-functional device capabilities are needed in Amigo both for selecting a device based on the capabilities it provides, and also for adapting services to the device nodes.

In the following, the information gathered in steps 1 and 2 serves as an introduction for the particular domain in this document. The lists in step 3 are not included into document, while only the result and rationale from step 4 are presented. Step 5 will follow later that is related with the definition of the properties or slots for the identified classes. Before reaching this pont, all the Amigo domains should be carefully studied, while ontologies adopted by them should be identified. Because each of the domains is first analysed separately, these results may have some overlapping description material in the domain-specific classifications.

2.1 Mobile domain

Here we are focusing on vocabularies related to various mobile platforms. Examples of such mobile platforms could be a mobile phone, PDA, laptop, smartcard etc. The significance of mobile domain in daily home is lower compared with for example the domotic or CE domains. Nevertheless, there are cases where mobile domain information can be used in Amigo such as the following:

- Communication services provided by each mobile phone need to be associated with its user for Amigo system.
- A washing machine alerts the user that water supply has been turned off during its program. Amigo system tries to alarm the person that started the machine. If no one is at home, it uses mobile phone to contact the specific user.
- A mobile phone/laptop enters Amigo home (local wireless connection) and announces its presence. Amigo registers some of its capabilities (restricted by security and privacy), as available Amigo services associated with the user.
- When the TV is in use, the Amigo systems use the mobile phone via WLAN as next best choice to show an urgent message to its user.
- There are cases where the user that participates to game/infotainment service sessions exits the Amigo house and he/she wants to continue using the specific service via the mobile phone. The Amigo system downloads the content/service to the phone. The Amigo service makes use of the context information concerning the mobile device hardware (e.g. the display size and colour depth) and its software capabilities (e.g. J2ME version) for downloading and adapting the service to the mobile node.
- User watches pictures, video, etc, on the mobile phone and wants to switch display to TV.
- Based on context information the Amigo system controls the profile set in mobile phone (set phone to silent mode or no work related calls accepted at home).
- The mobile phone is used as means of providing the location information of the user (subject to security and privacy restrictions), either by a service residing in the mobile phone or in the Amigo home.

2.1.1 Overview of available classifications for the mobile domain

Open Mobile Alliance (OMA, formerly known as the WAP Forum) is developing open technical specifications that, when combined, produce enablers for creating useful mobile services. The problem is that most of the capabilities of these enablers may be too application-specific, and thus not very useful from the Amigo point of view.

There are several classifications for describing the non-functional properties of mobile devices. There is a context initiative by W3C defining a set of attributes that characterise the capabilities of the access mechanism, the preferences of the user and other aspects of the context data concerning the web page to be delivered. Part of this work is integrated into the Composite Capability/Preference Profiles (CC/PP) standard [CCPP].

OMA has defined a User Agent Profile as an implementation of CC/PP for WAP-enabled mobile terminals, integrating mobile technologies with those of the Web [OMA05].

WURFL is a free, open source project that provides an alternative source of information to UAProf. It aims to provide a comprehensive view of device information, and contains information for 4500 variants of devices. Because WURFL is open source, anyone can edit/add device information, and not just device manufacturers. WURFL provides its own XML format for device characteristics description.

The foundation of intelligent Physical Agents has defined a FIPA -device ontology specification [FIPA91].

In general, these classifications enumerate technical properties of devices and could be formed as specialisations of more general taxonomies covering also PC and CE domains. The FIPA device ontology is a good starting point for the design of a device ontology. W3C is possibly also moving towards using OWL for vocabularies concerning CC/PP attributes.

2.1.2 Analysis of the mobile domain for Amigo

A set of taxonomies for mobile domain can be identified for further analysis:

A. Taxonomy for typical functional capabilities of mobile devices, provided as software services to be used in modelling of services for composability.

B. Taxonomy for mobile devices and platforms that can be used to define device context in the mobile domain.

C. Taxonomy for mobile content that is related to input and output information provided and required by services.

D. Taxonomy for context information that is specific to mobile devices related to context domain.

E. Taxonomy for QoS characteristics can be also considered, but it is not elaborated yet for mobile domain. It is mostly related to usability of service for the user because of restrictions of mobile devices and selection of most suitable communication channels when e.g. hand-over is needed because of mobility.

A. When classifying the functional capabilities, we focus on those capabilities, provided by mobile devices that can be used for service composition in Amigo. This involves means for communication with the user, provisioning of applications and media into the device, or control of the mobile device to integrate and adapt it with Amigo home. Communication with users by calling and messaging is an inherent capability of the mobile domain. Similarly to the work on Amigo Intelligent Services, the standardization in mobile domain focuses on providing a standard based set of application enablers, e.g. specific capabilities supporting development of mobile games. One of such enablers is management of user communities for games where gamers can upload their scores and participate in competitions against other players.

```
Capability
    UserCommunication
        Calling
            VoiceCall
            VideoCall
        Messaging
            MMS
            MultimediaMessaging
            SMS
            TextMessaging
    ContextManagement
```

ContextDetection

ContextStorage

DeviceManagement

ApplicationSupport

SessionManagement

ApplicationConnectivity

UserCommunity

SchedulingAndTiming

MeteringAndLogging

BillingModelSelection

ContentManagement

ContentStorage

B. The device capability profiles provide background for the mobile platform taxonomy. A common problem with the various standards considered is that they mostly describe only the variable set of capabilities of device, assuming a standard set of capabilities for the underlying platform technology. In Amigo such assumptions should be explicit. The selected taxonomy is based on various more or less standard device property descriptions but it tries to be more generic in order to cover also implicit information left out of those classifications. Each class in this taxonomy is a specialisation of the class that is higher in the hierarchy. For clarity, the taxonomy is restricted to three levels in which the first level defines a role of the platform, the second is a sub-classification of platforms provided in this role and the third level is usually a specific technology with possibly several instances.

MobilePlatform

MobileDevice

Laptop

MobilePhone

PDA

Accessory

Camera

Display

Handsfree

Keyboard

Microphone

Printer

Speaker

CommunicationPlatform

CommunicationProtocol

CellularNetworking

Cellular2G

Cellular3G

EDGE

- *WirelessNetworking*
    - *Bluetooth*
    - *Wlan*
- *ServiceArchitecture*
    - *WebServices*

HardwarePlatform

- ...

SoftwarePlatform

- *MobileMiddleware*
    - *FIPA*
    - *MUPE*
- *OS*
    - *Linux*
    - *Symbian*
    - *Windows*
- *ScriptEngine*
- *UIPlatform*
    - *Browser*
- *Bytecode*
    - *Java*

The above taxonomy provides a vocabulary based on which information related to mobile platforms can be represented by referring to the specific mobile device class (laptop, mobile phone or PDA or an accessory attached to one of those). This approach is selected for the above taxonomy because much of the knowledge in mobile platforms is usually implicit for different classes of devices, i.e., the device capabilities are not defined explicitly, but can be inferred from the class of the device. Furthermore, a mobile platform can be defined by using vocabulary of specific communication-, hardware-, or software-platforms used in mobile devices.

A partial draft of low level ontology based on the mobile platform taxonomy is illustrated in Figure 2.

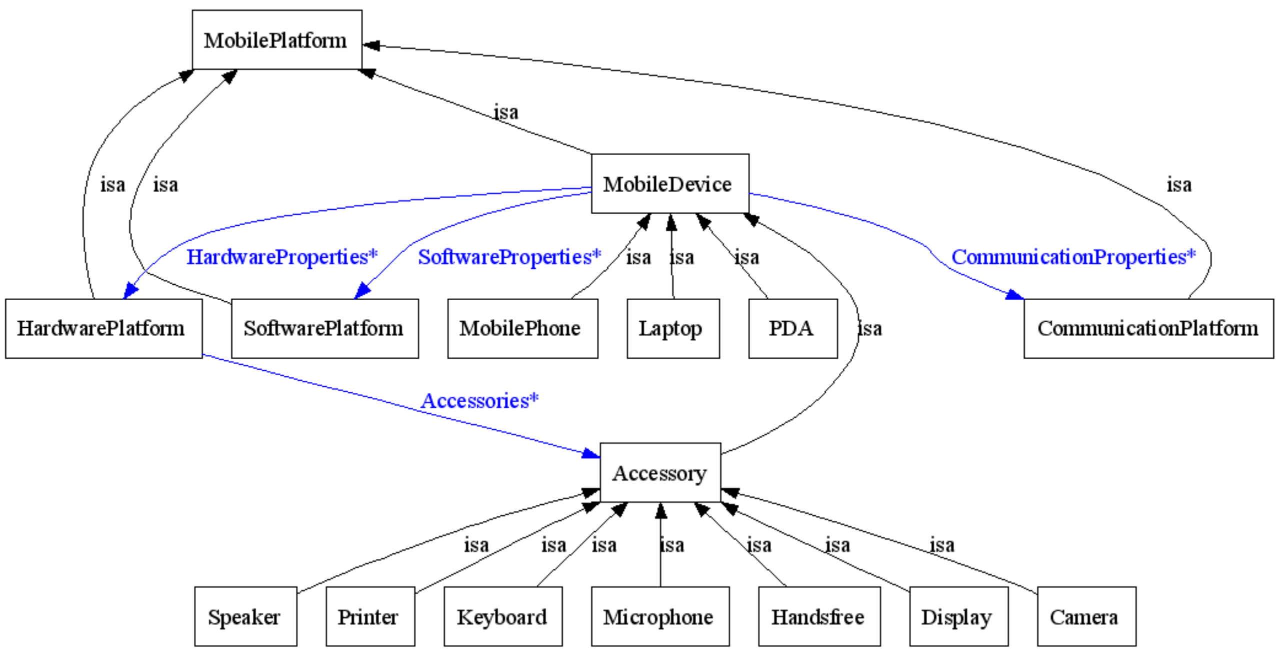

Figure 2 Vocabulary ontology based on the mobile platform taxonomy

Based on this ontology one can describe for example only the communication properties of a mobile platform, or define classes of mobile devices with restrictions on typical hardware, communication and software platform properties. Properties of a mobile phone model can be specified as an individual of class MobilePhone with specific properties. Similarly this ontology can be extended for example with typical capabilities provided by a class of MobileDevice if necessary.

C. The types of content in mobile domain are related to CE domain. Typical to mobile domain is that many applications are fixed in device but others, like Java games, are downloaded as content to device. Typical application data in mobile domain is related to contact and calendar information.

MobileContent

- *Application*
  - *Game*
  - *Calendar*
  - *Phonebook*
  - *CameraImaging*
  - *EmailClient*
- *ApplicationData*
  - *Bookmark*
  - *CalendarEntry*
  - *ContactInformation*
- *Media*
  - *Audio*
  - *Image*
  - *Video*
  - *Text*

Stream

Still

An individual of media class can belong several of media types (e.g. audio stream). A media stream can be played back before it is downloaded entirely into the device.

D. There are several types of context related information used in the mobile domain. Environment in which a mobile device and its user are can be used for purposes of instructing a service to adapt it by using larger fonts or provide backlighting to device display in poor visibility. Location of mobile device can be used to select the services that are closest to the user. Current mobile phones do not yet contain sensors for measuring environmental parameters and accurate location. Thus, physical conditions are classified into various profiles from which a user can select the most suitable for managing the various features of mobile devices. The relevant taxonomy is described in detail in the Physical Domain Context Ontology provided in chapter 2.5.

The device context which consists of the properties of the mobile platform is mostly used to adapt applications to the limitations of device.

2.2 PC domain

The PC domain is related to the “classical” view that we have about computers. Personal computers, Web cameras, peripherals such as printers, scanners, etc, are included in this domain. The nature of the resources of these devices can widely vary (e.g., in terms of connectivity, processing power, UI etc.). The role of PC domain in our every-day life is mainly related to storing, accessing and processing information. Relevant data may range from management information such as individual home preferences to access rights to multimedia content.

Subsequently, are some examples extracted from Amigo scenarios, where the PC domain is implicitly or explicitly involved:

- Amigo may download some specified content to a personal device such as game, photos, music.
- The list of video games, records and the preferences of each user are stored and managed by Amigo.
- Amigo takes care of inappropriate video games and asks for parental permission if necessary.
- Amigo can set-up a video-conference between two houses.

2.2.1 Overview of available classifications for the PC domain

The FIPA specifications include device ontology [FIPA91]. Work in FIPA device ontology is based on [CCPP] and [UAProf], thus aiming mainly to the mobile (WAP) domain. However, as domains overlap in many areas, some concepts and relations can be reused for the Amigo purposes. In this sense, device descriptions provided by FIPA specifications which involve hardware, connection types, user interfaces, resource/device capabilities (such as screen, resolution, memory, memory type) and software properties specification, may be useful to provide a reasoning framework for device, application, service and content selection. Moreover, [FIPA79] treats agent software integration and provides an abstraction for basic services of software systems, which are intrinsically linked to the PC domain.

Device Independence initiative [DIP03] focuses on making the Web accessible anytime and anyhow, in particular by supporting many access mechanisms (including mobile and personal devices, that can provide access anytime) and many usage modes (including visual and audio ones, that can provide access anyhow). Existing devices that are commonly used to access the Web include PCs, PDAs, web-enabled phones, and interactive TVs. This provides an intrinsic description of configurations for the content adaptation process, a process necessary to present content in a wide variety of devices which viable in order to achieve the Amigo objectives. Content is more related to the CE domain but it is also quite important for the PC domain, since personal devices are increasingly attractive in multimedia content presentation, distribution and conversion.

In [SKWIH], a general framework for device and service description is proposed. It is useful from the point of view of checking very general principles of device description. However, it lacks specificity. An example of a specific description of devices appears in [BPRC04], together with a framework for semantic descriptions of devices. Several abstract references can be taken into account for classification purposes.

Swoogle [Swoogle] provides a search engine for some existing ontologies. Although this source is far from being complete it may provide some useful classification examples and items to fulfil the lack of domain specific instantiations presented by the above sources. Many of the PC domain devices that can be found in an Amigo home are listed and classified here.

2.2.2 Analysis of the PC domain for Amigo

A set of useful taxonomies for the PC domain can be identified as follows:

A. Taxonomy for typical functional capabilities of personal devices.

B. Taxonomy of PC devices into processing units and peripherals.

C. Taxonomy for PC devices and platforms which can be used as context information.

D. Taxonomy of QoS properties related to PC device classes.

A. PC domain capability classification intends to organise software agents and services. This is a huge task, and thus, some simplification should be made focusing on the achievement of Amigo's objectives.

PCDomainCapabilities

- *CommunicationCapability*
    - *AudioVisualCommunication*
        - *Videoconference*
            - *H.320*
            - *H.323*
            - *RTSP*
    - *Messaging*
        - *Email*
        - *Messenger*
    - *VoiceCommunication*
        - *ModemCall*
        - *VoIP*
- *PCDeviceControlCapability*
    - *PowerControl*

        - *ResetDevice*
        - *RestartDevice*
    - *IOCapability*
        - *InputCapability*
        - *OutputCapability*
            - *Printing*
                - *Spooling*
    - *ControlCapability*
        - *ApplicationControlCapability*
            - *Connection*
                - *Close*
                - *Connect*
            - *Eventing*
                - *SubscribeToEvent*
                - *UnsubscribeToEvent*
            - *IOControl*
                - *BindIOObject*
                - *DeregisterIOObject*
                - *RegisterIOObject*
            - *InvokeAction*
            - *State*
                - *RetrieveState*
                - *StoreState*
        - *ProcessingControlCapability*
            - *Init*
            - *SetProcessingPriority*
            - *Resume*
            - *Suspend*
            - *Terminate*
    - *UserInterfaceCapability*
        - *Browsing*

B. Classification of devices by functionality is useful in order to define common properties or relations to classes. Desktop PCs, PDAs and Laptops can run processes and therefore they will have similar properties. A screen, a printer and a speaker can be used to output information.

- *Device*
    - *Peripheral*
        - *DuplexDevice*
            - *Fax*

Modem

TouchScreen

InputDevice

Joystick

Keyboard

Microphone

Mouse

Scanner

Webcam

OutputDevice

Printer

Screen

Speaker

StorageDevice

PortableStorage

CD-DVDReaderWriter

PortableHDD

USBPenDrive

StaticStorage

HDD

FloppyDiskUnit

ZipUnit

ProcessingUnit

DesktopPC

Laptop

PDA

PocketPC

Palm

C. Similar classification to software, hardware and connectivity platforms as in Mobile domain can be used also in PC domain:

PCPlatform

SoftwarePlatform

OperatingSystem

Linux

Solaris

Unix

Windows

MacOS

VirtualMachine

- *AgentPlatform* (continued)
  - *JVM*
- *AgentPlatform*
  - *AgentConnection*
    - *Port*
    - *Protocol*
  - *AgentIIO*
    - *InternalRequisites*
    - *PeripheralRequisites*
  - *AgentRuntimeQuality*
    - *MinimumMemory*
    - *MinimumProcessor*
    - *Platform*

ConnectivityPlatform

- *LinkProtocol*
  - *Bluetooth*
  - *IrDa*
  - *IEEE1394*
  - *MIDI-GamePort*
  - *PS/2*
  - *Parallel*
  - *RS232*
  - *USB*
  - *WiFi*

D. PC domain QoS feature taxonomy provides a background classification for the property assignation to device classes. Among others, it allows reasoning and decision support in cases such as the determination about whether specific content can be presented on particular devices (e.g., if some multimedia content such as movie can be streamed from TV to PC).

PCDomainQoS

- *ProcessingQuality*
  - *MemoryCapacity*
  - *ProcessingCapacity*
- *StorageQuality*
  - *StorageCapacity*
- *PrintingQuality*
  - *PrintingFormat*
  - *ColourTreatment*
  - *PagesPerMinute*
  - *PrinterResolution*
  - *PrinterQueueCapacity*

- *ScreenQuality*
  - *ColourDepth*
  - *ScreenResolution*
  - *ScreenSize*
- *VideoCaptureQuality*
  - *VideoEncoding*
  - *FramesPerSecond*
  - *ImageSize*
  - *VideoResolution*
- *ScanningQuality*
  - *ScannerColourDepth*
  - *ScannerResolution*
  - *Speed*
- *KeyboardQuality*
  - *FunctionalExtraKeys*
  - *WindowsExtraKeys*
  - *MultimediaExtraKeys*
  - *Layout*
    - *EastAsianLayout*
    - *NonRomanAlphabeticScriptLayout*
    - *RomanScriptLayout*
      - *AZERTY*
      - *QWERTY*
      - *QWERTZ*
      - *QZERTY*
- *JoystickQuality*
  - *NumberOfAnalogueChannels*
  - *NumberOfDigitalChannels*
  - *VibrationCapability*
- *AudioQuality*
  - *NumberOfChannels*
  - *OutputAudioPower*
- *MouseQuality*
  - *BallMouse*
  - *OpticMouse*
  - *TrackBallMouse*

Most of these features are strongly related to the corresponding device functionality class (see Figure 3). Others enable possible relations between devices demonstrating different functionalities (i.e. connectivity features, which can answer questions such as: ”Can this keyboard be connected directly to this PC?”).

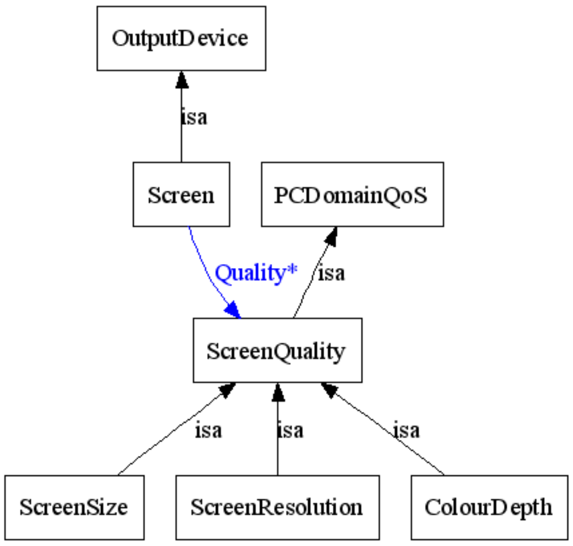

Figure 3 Relation between device and QoS vocabularies.

Some other classifications closely related with the PC domain and also important for efficient service discovery are: the context taxonomy with regards to the location of devices in the home (e.g. nearest printer) and also the security and privacy taxonomy, which would enable for example reasoning about access rights for particular applications in specific devices of the PC domain.

2.3 CE domain

Analysis of the CE domain in Amigo concerns audio, video and other entertainment devices in the home. The interaction of home individuals with this domain mainly takes place during their leisure time, thus making the CE domain a very attractive, actively utilized and potentially profitable one. Plenty of CE devices such as HiFi systems, TV screens, video consoles, sound speakers, etc, can be found in our homes. The entertainment content which is displayed and exchanged between those CE devices is quite attractive and important from the user’s point of view. This content includes songs, films, still photos, etc, and can be seen as indispensable attribute of the CE domain. In addition, a many of the data linked to this content (such as size, bit rate, and parental permissions) can affect the user’s experience directly or indirectly.
Here are some examples extracted from the Amigo scenarios, where the CE domain is implicitly or explicitly involved:

- When someone wakes up, Amigo starts his/her day by offering some personalized content (music, news, radio host).
- Content can be switched between various heterogeneous devices across the home, so that the user is followed by the content.
- In the Amigo home, almost every room/corridor/place can be equipped with a screen (and probably with some speakers).
- A user may start a karaoke session assisted by the Amigo system (example of another “not classical” entertainment device/service).
- The list of video games, records and preferences of each user are stored and managed by the Amigo system.

- The Amigo system takes care of inappropriate video games and asks for parental permission if necessary.
- Audio and video is adapted to the infotainment experience throughout the room.
- The video game is presented on a large wall.
- Different game devices of multiple players are integrated with their preferences.
- When watching a movie, sound and video are adapted according to user preferences and current context.
- Amigo selects the games that players in different houses like to play together, and enables then to experience face-to-face interaction during the game.
- Individuals in different houses can watch TV together, participating in a videoconference.

2.3.1 Overview of available classifications for the CE domain

The TV-Anytime Forum [TvAnytime] is in progress of defining specifications that will enable applications to exploit local persistent storage in consumer electronics platforms. Useful ideas and knowledge can be used from this source for the CE domain and the content taxonomies. Within the TV-Anytime environment, the most visible parts of metadata are the attractors/descriptors used, e.g. in Electronic Program Guides (EPG), or in Web pages to describe content. This is the information that consumers or intelligent agents will use to search and select content available from a variety of internal and external sources. Another important set of metadata consists of user preferences descriptions, representing user consumption habits, and defining other information (e.g. demographics models) for targeting a specific audience. However this source is more broadcasting oriented (i.e. TV-world specific), so it does not lie directly within the Amigo scope.

The FIPA specifications include device ontology [FIPA01]. Work in FIPA device ontology is based on [CCPP] and [UAProf] and it mainly focused on the mobile (WAP) domain. However due to the lack of sources for the CE domain (especially referred to devices), relevant ideas can quite useful.

In addition, the FIPA specification made an effort to AV Entertainment [FIPAAV]. This specification describes the assessment of FIPA specifications against a prototypical Audio/Video Broadcasting and Entertainment application. Some descriptions in [FIPAAV] about AV Objects, parental rating, etc, can inspire the design of the Amigo ontologies.

Device Independence initiative [DIP03] focuses on making the Web accessible anytime and anyhow, in particular by supporting various access mechanisms including mobile and personal devices, which can provide visual and auditory access modes. The information extracted from this source can be useful since there are some CE devices, which can be used to access the Internet (e.g. TV). Hence some descriptions in the Device Independence initiative can be taken into consideration.

Some classifications with regards to multimedia content description [MPEG7, MPEG21, [TV Anytime] have been designed. The MPEG standards form an evolving set of standards for video and audio compression. MPEG-7 technology covers the most recent developments in multimedia search and retrieval, designed to standardise the description of multimedia content supporting a wide range of applications and supporting media. The semantic metadata included in MPEG7 about content and subject description is rather out of the Amigo scope, but the way the information is described and organized in this standard can be of great help for the Amigo purposes.

In [TsPo04] a systematic methodology that allows coupling of OWL with MPEG-7 and TV-Anytime is described. First, domain-specific knowledge is transparently integrated into MPEG-

7 and TV-Anytime, thus enhancing multimedia retrieval performance. Second, multimedia content and domain-specific information described in OWL is mapped into MPEG-7 and TV-Anytime and vice versa. The methodology greatly facilitates information integration, retrieval and interoperability in Web application environments. An MPEG-7 compatible Retrieval API is also described that has been developed for expressing powerful retrieval queries on multimedia content utilizing domain specific knowledge and application specific metadata knowledge. The development of an OWL ontology that fully captures the metadata model has been defined in the Semantic Part of the MPEG-7 MDS.

Finally, Swoogle [Swoogle] provides a search engine for some existing ontologies. It is far from complete, however may provide some useful classification examples and item listings.

2.3.2 Analysis of the CE domain for Amigo

For the CE domain some possible classifications can be:

A. Taxonomy for CE devices

B. Taxonomy for typical functional capabilities of CE devices

C. Taxonomy for multimedia content

A. The consumer electronics devices are typically classified into several subgroups based on the type of the media they can handle:

- *ConsumerElectronicsDevice*
  - *AudioDevice*
    - *CDPlayer*
    - *DVDPlayer-Recorder*
    - *Headphones*
    - *HiFi*
    - *PortableDVDPlayer*
    - *PortableMP3Player*
    - *Speaker*
    - *Microphone*
  - *GamingDevice*
    - *GameController-Joystick-Pad*
    - *VideoConsole*
    - *GameMemoryUnit*
  - *ImageDevice*
    - *(DVDPlayer-Recorder)...*
    - *DigitalPhotoCamera*
    - *(PortableDVDPlayer)...*
    - *(HandheldTVPlayer)...*
  - *PortableDevice*
    - *DigitalCamcorder*
    - *(DigitalPhotoCamera)...*
    - *(PortableDVDPlayer)...*

(PortableMP3Player)...

VideoDevice

(DVDPlayer-Recorder)...

(DigitalCamcorder)...

(PortableDVDPlayer)...

STBDecoder

TV

(HandheldTVPlayer)...

VCR

Many of the modern devices belong to several of those subgroups. For example, digital photo cameras can also record live video, some set top boxes may download games and future gaming devices may render stream digital video from the Internet. These devices are represented in the taxonomy enclosed in brackets. This has to be kept in mind creating the particular instances of the classes.

B. The fragment from Amigo scenario *'Maria wakes up and walks towards the bathroom, with the music following her'* may imply various reasoning with regards to the following:

- The location of various devices in Amigo home for discovering the CE devices in the room where Maria enters.
- Selecting the devices that are able to play audio.
- Deciding whether the services of these devices support starting playback at a given point in the song (halfway playback).
- Knowing which audio formats (and which characteristics such as bitrate) the devices support and being able to select the device providing the best quality of sound.

The halfway playback can be achieved for example, if the target device supports *play* and *seek* capability. Such capabilities of CE devices can be classified by considering the roles that devices play when handling the media, i.e.:

- This device can control other devices (a control point).
- This device can retrieve and store content (a media server).
- This device can playback content (a renderer).

A draft vocabulary for such capabilities could be:

ConsumerElectronicsCapability

MediaControlPoint

PlaybackControl

Pause

Play

Seek

Stop

MediaRenderer

GetContentFrom

RenderingControl

BrightnessControl

ContrastControl

- *VolumeControl*

- *MediaServer*
  - *BrowseContent*
  - *Convert*
  - *RetrieveContent*
  - *Search*
  - *SendContentTo*
  - *StoreContent*

C. QoS aspects in the service discovery may rise some questions related to the format of used multimedia content in particular situation/context, i.e., which format for this content is the best suitable for the current state of the network. Various types of multimedia content may lead to the following classification:

- *MultimediaContentType*
  - *ContentSize*
  - *Bitrate*
  - *NumberOfChannels*
  - *Resolution*
    - *SpatialResolution*
    - *TemporalResolution*
  - *MediaFormat*
    - *AudioFormat*
      - *ADPCM*
      - *CS-ACELP*
      - *G.711*
      - *MP3*
      - *OGG*
      - *PCM*
      - *WAV*
      - *Mono-Stereo*
    - *ImageFormat*
      - *BMP*
      - *GIF*
      - *JPEG*
      - *TIFF*
    - *TextFormat*
      - *Plain*
      - *RTF*
    - *VideoFormat*
      - *AVI*
      - *DivX*

MPEG2

MPEG4

WMV

XVid

2.4 Home automation (Domotic domain)

The domotic domain in Amigo is related to identifying the requirements of home automation devices to Amigo system. Some examples of such devices could be a lighting system, a washing machine, a gas sensor, etc.

Considering the refined Amigo scenarios described in deliverable D1.1 [Amigo-D1.1], some examples of these requirements from domotic systems can be easily extracted:

- When light and sound effects are important for video games, Amigo adapts the home environment by presenting lights.
- When Maria and Jerry are watching a film, Amigo adapts the home environment and creates an all-surround sound, light and video ambiance.
- Amigo recognizes people at the front and patio doors of the house and opens the appropriate door(s) for them.
- Amigo can take over housekeeping tasks.
- Starting and stopping the working of appliances at a desired time with the correct settings for session duration, power supply, input material dosage, temperature, etc.
- Amigo can even detect the presence of inappropriate objects in or near the appliances.
- Amigo, as a kitchen chef, downloads recipes and cooking programs to the kitchen.

The diversity and heterogeneity of domotic elements enforce to define a classification of the services provided by these systems, in order to be used within Amigo system.

2.4.1 Overview of available classifications for the domotic domain

Due to the great heterogeneity of devices that can be found in the domotic domain, there is no available classification covering the entire home automation world. As a starting point, domotic devices can use the FIPA device ontology.

2.4.2 Analysis of the domotic domain for Amigo

A set of preliminary classifications for the domotic domain was selected to be further studied:

A. Taxonomy for the domotic platform, describing a scope of domotic devices and their capabilities.

B. Taxonomy for domotic states, classifying the diverse states a domotic device can be on.

C. Taxonomy for domotic events, describing the events that can occur and be raised in a domotic environment.

A. Trying to classify the diverse domotic devices, we focus on those devices that could be included in Amigo scenarios and demonstrators. Any device has two basic properties: an identifier, which makes it unique, and a state, which describes how the device currently is.

The domotic device states are described and analyzed later in this document. The first classification of domotic devices can be: sensors, actuators and appliances.

- *Device*
  - *Sensor*
    - *Detector*
      - *Gas detector*
      - *Water detector*
      - *Presence detector*
    - *Measuring sensor*
      - *Temperature sensor*
      - *Light sensor*
      - *Sound sensor*
      - *Humidity sensor*
  - *Actuator*
    - *Switch*
      - *Binary light*
      - *Watering*
      - *Door lock*
      - *Window lock*
    - *Regulator*
      - *Regulable light*
    - *Valve*
      - *Water valve*
      - *Gas valve*
    - *Engine*
      - *Blind*
      - *Curtain*
      - *Gate*
  - *Appliance*
    - *Programmable appliance*
      - *Oven*
      - *Washing machine*
      - *Dishwasher*
    - *Refrigerator*

Sensors can either detect environmental changes or measure environmental conditions, providing relevant information. The first sensor category can notify about such changes, while the second provides specific values.

Actuators, on the contrary, can interact with the environment changing some conditions. There are several types of actuators. We can identify switches, regulators, valves and engines. Switches can change a binary state (i.e. on/off, or open/close). Some examples of switches are binary lights, locks, polarized windowpanes, etc. Regulators adjust the parameters of devices that form continuous variables, such as the illumination level of a configurable light. Valves are similar to switches but don't control electrical devices. Finally, domotic engines allow to open a gate or rise the blinds. Some actuators can be scheduled, setting their start time and duration or a periodicity. In the case of engines, the duration of each movement (required up and down time to open or close a blind) can also be set.

Appliances or white goods are another subset of domotic devices. Some of them can be scheduled (set a start and end time, the temperature of the oven according to a recipe, the washing machine program etc.).

B. There are several states a domotic device can be on. The next classification shows the different states that can support a domotic device. Some devices will support only a subset of them, whereas other devices can support all of them.

State

- *Off-line*
  - *Disconnected*
  - *Communication error*
- *On-line*
  - *On*
  - *Off*
- *Alarm*
  - *Scheduled*
  - *Paused*
  - *Fault*

A domotic device can be Off-Line or On-Line. The off-line state doesn't allow us to control the device and can occur either if the device is disconnected or if the communication fails. Some on-line states can be identified: the device can be working or stopped (a washing machine), can be on alarm (the refrigerator is over its recommended temperature), scheduled, paused or in fault.

C. Trying to analyze and classify the events that can occur within a domotic installation, we can summarize them in the following ones: alarm, fault and state change:

- An alarm event occurs when a device identifies a problem: a red sock is detected in the washing machine with the white clothes or a water leakage is discovered by a water detector in the kitchen.
- A fault event is raised when a device detects an internal malfunction: the refrigerator temperature is not adequate.
- A device notifies the appropriate equipment sending a state change event, when it modifies its own state, e.g. when the washing machine program has finished, it notifies the power supply control unit to switch from on to off.

2.5 The discussion on non-functional aspects for service modelling

To enable context- and QoS-aware service discovery some context/QoS information can be statically included in service descriptions or can be dynamically provided (i.e accessed/notified) to improve the quality of retrieved results in the matching process of service discovery. The vocabulary ontology on context and QoS for Amigo-aware services description and discovery is discussed in the following subsections.

2.5.1 Context-awareness

As stated, context awareness is an important attribute of Amigo home domain. As a plethora of heterogeneous services, devices and networks are incorporated in the home environment, while users get in and out of the system constantly, well updated and accurate context information is required. In case this information is available, the Amigo system is made aware of the user state, environment and activities, and is able to select the most appropriate way to address the user requirements under the current conditions using efficiently the resources that are available. On the other hand, based on such a system, Amigo users can identify and locate all entities in their home environment and are able to communicate, interact, configure and use them on demand.

Making traditional services context aware is a challenging task as the appropriate middleware support is absent in most intelligent systems. This especially stands for the home domain, where context-aware supportive infrastructures are not yet available in the real world. Such infrastructures require appropriate technologies to represent, collect, interpret, access, manipulate and disseminate context information. Here we continue an analysis on context ontology which would fit the needs of Amigo home environment. The context ontology description language is discussed in deliverable D3.1b [Amigo-D3.1b].

2.5.1.1 Overview of available classifications and context related ontologies

In this section an overview of the main research work on the design of context ontologies is provided. It should be mentioned, that even though vocabularies for describing context presented in relevant literature are enough for specific application domains, context ontology description languages are absent or very simplified.

In [GuPZ04], context ontologies are defined based on OWL. A generic Upper Context Ontology is introduced, while some domain-specific Lower Context Ontologies are proposed. The Upper Ontology is based on four main entities: ComputationalEntity, Location, Person and Activity. Various domains are identified, such as the Home Domain, the Vehicle Domain and the Office Domain. In this approach, information is classified in Direct and Indirect Context, depending on whether context data is obtained directly from a context source/provider or not, while four types of quality of context parameters are identified: accuracy, resolution, certainty and freshness. This research team has also worked on incorporating some reasoning techniques in context ontologies that can be used to enhance context–awareness [WZGP03]. Evaluation results indicate that the performance of the reasoning mechanisms used depends on the volume of context information, the complexity of reasoning rules and the CPU speed. Nevertheless, the provided Person ontology is quite limited. Furthermore, the Time ontology, an ontology which is vital for the Amigo platform is absent.

In [StLF03], an Aspect-Scale-Context (ASC) model is presented, from which a Context Ontology Language (CoOL) is defined. CoOL consists of two main parts: CoOL Core, which is based on OWL and F-Logic, and CoOL Integration which is a collection of schema and protocol extensions. F-Logic is used as a knowledge representation language. In the proposed ASC model, each aspect aggregates one or more scales, whereas each scale aggregates one or more pieces of context information. Furthermore, an extension of OWL-S is presented, which incorporates concepts necessary for service context ontologies by introducing the Service Context class as a subclass of the general Service class. In this framework, a system architecture has also been designed and developed, in order to validate the ASC model and the proposed ontology schemes. Even though the proposed model can be used to establish context-awareness, while it supports efficient context information exchange, the ontology analysis focuses only on specific applications.

An Agent-based Context-Aware Infrastructure (ACAI) is presented in [KhKa05]. ACAI provides a unified context representation scheme, while it supports context management, composition, inference and dissemination. The proposed infrastructure consists of three layers: a sensing layer, a context service layer and an application layer. Central entities to the proposed ACAI's multi-agent system are the Context Management Agent, the Coordinator Agent and the Ontology Agent, that handle context management, coordination and semantic interoperability respectively. The key elements of the proposed ontology are: Location, Actor, Network, Service and Action. In the context model used, a hierarchy is identified based on the depth of information provided. Thus, the root level contains the most generic concepts of context (*ContextView*) and as we step into sublevels, more details about each concept is provided. This work focuses on designing a context-aware service provision infrastructure, while the proposed context ontologies are quite limited with respect to representing a high level view of the entire context information domain.

Another context ontology is presented in [PBWG04]. The proposed model is designed especially for Ambient Intelligence environments. Several requirements are identified with regards to context models and systems for AmI services, such as: application adaptivity support, resource awareness, mobile service support, semantic service discovery, code generation, and context-aware user interface provision. The generic Context Ontology consists of four sub-ontologies: (i) user, (ii) environment, (iii) platform, and (iv) service ontologies. The latter is based exclusively on OWL-S. Like most other research groups, this team hasn't addressed the design and development of a context ontology language.

In [ChFi03], a pervasive context-aware computing architecture is presented named CoBrA (Context Broker Architecture). The proposed infrastructure is based on Brokers, which retrieve context information from devices, agents or other sources. The main responsibilities of these Brokers are: to obtain context information, to reason about the context, to contribute to spreading context information among agents and to ensure the user privacy. A key issue in this framework is the fact that the entire communication, interworking and representation depend fully on ontologies. Thus, focusing on the university campus domain, a set of common ontologies based on OWL has been used that enables knowledge sharing and ontology reasoning. Main classes in these ontologies are: Person, Place and Intention. The context ontology vocabulary produced by this team, it is not adequate for Amigo as it is very application specific, furthermore no context ontology language has been provided.

A research effort that focused on Quality of Context is described in [BuKS03].In this approach, Quality of Context (QoC) is "*… any information that describes the quality of information that is used as context …*". Thus, QoC refers to information and not to the process or the hardware component that possibly provides the information. The most important QoC parameters identified are: precision, probability of correctness, trust-worthiness, resolution and up-to-dateness. A relation between QoS, QoC and QoD (Quality of Device) is also described. This work is quite interesting with regard to the representation of the quality of the context information. Nevertheless, as far as the context ontology vocabulary and language are concerned the semantics developed are limited.

2.5.1.2 Core context categories

Several context categories can be identified based on discussions on different domains and inspired by Amigo scenarios, as most important to meet the requirements of context-aware service discovery in Amigo aware home environment. These are:

- *User Domain.* This domain represents information that describes individuals in the home. User context domain may include the physical characteristics of person, personal information, application related information (e.g. email received, Web sites visited), preferences, activity, biological/psychological state, personality, etc. This information will facilitate the provision of services personalized for each individual in the home such as the downloading of his/her favorite game or selecting of appropriate music for some occasion or adapting of the right user interface, so as to be convenient and attractive for specific home inhabitants like children, adults, guests.

- *Physical Domain*. The classes of this domain aim at representing the physical parameters of the Amigo environment. These parameters include environmental, spatial and temporal properties, and have been introduced in order to cover the plethora of physical context information. Such parameters model, among others, the physical location of the user and the device, their temporal settings and the environment they are situated in. As an example, using this information, Amigo (as a doorkeeper) may send the photo of the guest in the nearest display or facilitate the "follow me" scenario when somebody would like to continue his/her game on the way.
- *Device Domain*. This domain is closely related to the four functional domains discussed in the previous sections. The properties of different devices that operate inside the Amigo home from the home automation, the consumer electronics, the mobile communications and the personal computing domains will play the role of context in the process of services discovery and selection to satisfy the user needs. Such aspects as available network interface, display properties, memory capacity, processing speed, etc. can be analysed here.
- *Object Domain*. This domain may include the representations of the non-living physical elements of the Amigo home (e.g. furniture, windows, doors).
- *Non-human Beings Domain*. The classes of this domain represent all living creatures that may exist in the Amigo home environment, apart from humans (e.g. pets and plants). This domain will be elaborated in our future work.

The more detailed analysis of User, Physical and Object domains is presented in the following sections.

2.5.1.3 The User domain context ontology

In this section the user domain context ontology is presented. In this initial approach an effort has been made to provide a complete context ontology for the user domain, which probably addresses more context parameters than those involved in the Amigo scenarios. However, whether this approach is best suited for Amigo or a less sophisticated context ontology version is more appropriate is still to be decided.

The classes identified in the proposed user domain context ontology are the following:

- *User*: The User class is the central class of this domain. It represents humans and holds relationships to all main classes of the user domain context ontology. The User class is also related to classes identified in other context domains (e.g. physical, device domain), such as the physical location, device used, etc. There are also multiple object properties relating two users. These properties represent relationships such as: studentOf, friendOf, motherOf, employeeOf, colleagueOf, etc.
- *Schedule*: The Schedule class captures all information related to the user’s agenda. It is a collection of scheduled Activities that are defined by type and participants using also spatial and temporal parameters. There is a one-to-one relationship between the Schedule and the User instances.
- *Activity*: The Activity class describes the activity performed by the user at a given time in the past, present or future. The Type attribute of this class is a string datatype property that describes the nature of the user activity. Examples of values for this Type property are: Walking, Running, Sitting, Sleeping, etc. As already stated, the Activity class is related to the user Schedule. It is also related to the Interval class and the Area class of the Physical Context domain described next, in the notion that each user’s activity is performed in a well defined Area and has a specific duration. It is noted here that the User and the Activity classes are related with two distinct object properties: the “performsActivity” property (User → domain, Activity → range) and the “involves-User” property (Activity → domain, User → range).

- *PersonalDetails:* The PersonalDetails class incorporates the personal information of a specific user that can be useful for the system or another user. This information includes the user's Name, Gender, Age, Address, Nationality, Country, City, Weight, Height, etc. All the fields above are registered as datatype properties of the PersonalDetails class, are optional and may be disclosed to the interested parties according to the user privacy settings. Each User has exactly one PersonalDetails instance.
- *Preferences:* The Preferences class is an abstract class that represents the preferences a user of the Amigo system may have. It is extended by the InterfacePrefs, ApplicationPrefs, InterestPrefs, QoSPrefs and OtherPrefs classes. The latter class may include more domain-specific preferences that cannot be classified under one of the prior four subclasses (e.g. food preferences). A user can have one or more preference profiles for each subclass, while their selection may depend on the current conditions and other user context information.
- *BiologicalState:* BiologicalState is the class introduced to represent the physiological status of the Amigo user. It includes attributes related to the user's health and his physical condition. Example properties under the BiologicalState class are: Health, Endurance, Adrenaline levels, HeartBeat, Fatigue, Hunger, Thirst, Sleepiness, Temperature, etc.
- *PsychologicalState*: The PsychologicalState class represents a collection of the psychological characteristics of the Amigo user. The psychological state of the user includes characteristics such as Fear, Anger, Joy, Worry, Compassion, Pride, Love, Hate, etc. and can be set in a static or dynamic (real-time) way. These characteristics are represented via the PsychologicalFeatures class and every instance of the PsychologicalState is formulated as a collection of such discrete features. The PsychologicalFeatures class is simple and includes a single datatype property (Type). The cardinality of the relationship between the PsychologicalState class and the PsychologicalFeatures class is 1:n.
- *Personality/Behavior:* The Personality/Behavior class incorporates all the aspects of the Amigo user's character. A person may be Shy, Stable, Optimistic, Calm, Kind, Intelligent, Open-Minded, etc. These characteristics may be updated periodically to reflect possible changes of the user's idiosyncrasy. They are formulated by the Type datatype property of the class PersonalityFeatures. The way these two classes are related is similar to the PsychologicalState class' case. The relationship is again 1:n.

The User domain context ontology is depicted in Figure 4

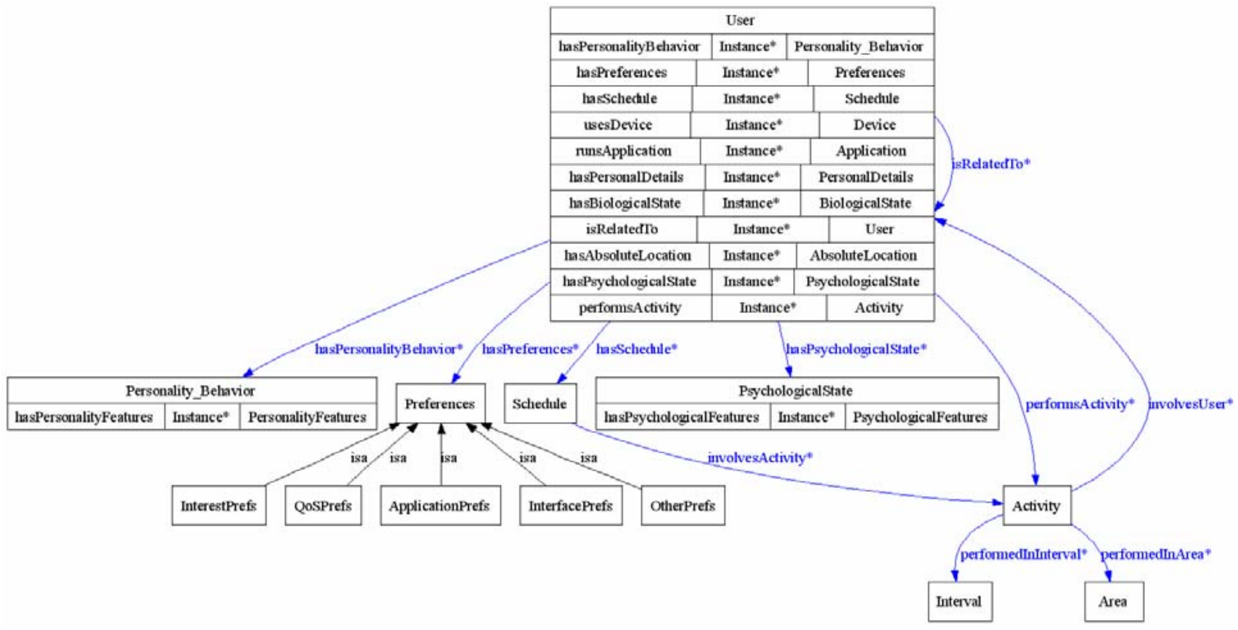
User
hasPersonalityBehavior | Instance* | Personality_Behavior
hasPreferences | Instance* | Preferences
hasSchedule | Instance* | Schedule
usesDevice | Instance* | Device
runsApplication | Instance* | Application
hasPersonalDetails | Instance* | PersonalDetails
hasBiologicalState | Instance* | BiologicalState
isRelatedTo | Instance* | User
hasAbsoluteLocation | Instance* | AbsoluteLocation
hasPsychologicalState | Instance* | PsychologicalState
performsActivity | Instance* | Activity
isRelatedTo*
hasPersonalityBehavior*
hasPreferences*
hasSchedule*
hasPsychologicalState*
Personality_Behavior
hasPersonalityFeatures | Instance* | PersonalityFeatures
Preferences
Schedule
PsychologicalState
hasPsychologicalFeatures | Instance* | PsychologicalFeatures
performsActivity*
involvesUser*
involvesActivity*
isa
InterestPrefs
QoSPrefs
ApplicationPrefs
InterfacePrefs
OtherPrefs
Activity
performedInInterval*
performedInArea*
Interval
Area

Figure 4 The User domain context ontology.

2.5.1.4 The Physical domain context ontology

In this section the physical domain context ontology is presented focusing on the Amigo home specific issues. This ontology incorporates the generic parameters that are related to the elements of the physical environment. It consists of three independent sub-ontologies that are described below. Spatial domain ontology is further visualized in Figure 5.

- Spatial Domain Ontology
  - *Space:* The Space class of the spatial domain context ontology is an abstract class that corresponds to any physical place. It holds an object property to the AbsoluteLocation class introduced later. Four subclasses of the Space class are defined and are presented below.
  - *AbsoluteLocation:* It represents the physical location of the user in terms of Longitude, Latitude and Altitude. Based on these attributes we can locate any context entity having physical substance. The AbsoluteLocation class is related to the *Distance* class that represents the distance of two entities of the physical world in (x,y,z) coordinates. Thus, the Distance class carries two object properties called fromAL and toAL (Distance → domain, range → AbsoluteLocation).
  - *ReferenceSystem*: Inside an AmI featured home it is necessary to locate objects based on their relative locations with regard to a specific reference point. Thus, the ReferenceSystem class has been introduced to represent the custom coordinates system from which the relative location is expressed. In most cases ReferenceSystems are tailored to the location and shape of selected physical objects in a specific Area. Each object of the ReferenceSystem class is defined by three data properties that indicate the directions of the three orthogonal reference axes (x,y,z). This class carries a mandatory object property called withZeroPointLocatedAt (ReferenceSystem → domain, range → AbsoluteLocation), which indicates the absolute location of the zero point (i.e. point (0,0,0)) of the custom ReferenceSystem.
  - *RelativeLocation*: Each physical object may have multiple relative locations with respect to the reference systems defined. These are expressed by instances of the RelativeLocation class that uniquely define a physical object's position

inside the Amigo home. There is a mandatory object property called basedOn (RelativeLocation → domain, range → ReferenceSystem) that indicates the reference system based on which the RelativeLocation is expressed. Each instance of the RelativeLocation class is defined by a triplet of coordinates (x,y,z) with regards to the three axes of the selected ReferenceSystem. Note that RelativeLocation is related to the AbsoluteLocation via two symmetric relationships (i.e., the correspondingToAL relationship with RelativeLocation → domain and range → AbsoluteLocation, and the correspondingToRL relationship with AbsoluteLocation → domain and range → RelativeLocation). These relationships are used to express RelativeLocation in AbsoluteLocation coordinates and vice versa. The RelativeLocation class is related to the Distance class, via FromRL-ToRL relationships (as in the AbsoluteLocation case). Thus, the Distance class carries two additional object properties called fromRL and toRL (Distance → domain, range → RelativeLocation).

  - *Country:* The Country is a subclass of the Space class that is used to identify the country in which specific coordinates are located (AbsoluteLocation). Each Country is uniquely identified by an international string identifier.
  - *City:* The City is a subclass of the Space class that is used to identify the city in which specific coordinates are located (AbsoluteLocation). Each City holds a cPartOfCountry object property (City → domain, Country → range).
  - *Area:* The Area class is modeled as a subclass of Space and represents a physical area located inside a City or a Country. It may correspond to an indoor or outdoor area. Each Area holds either a aPartOfCountry object property (Area → domain, Country → range) –this stands for the rural areas– or a aLocatedInCity object property (Area → domain, City → range) –this stands for the urban areas.
  - *Building:* The Building class is used to represent a physical building. As buildings are located inside an Area, this class holds a LocatedInArea object property (Building → domain, Area → range). It is related to the Room class, as every Room belongs to a building.
  - *Room:* The Room class is a subclass of the Area class and corresponds to a room located inside a building. It is identified by a string datatype property. Each Room holds a LocatedInBuilding object property (Room → domain, Building → range). The Room is critical for all Amigo home based scenarios.

- *Temporal Domain Ontology*
  - *Time:* The Time class is a collection of the temporal parameters that define a specific moment in time. It contains attributes such as TimeZone, Hour, Minute, Second, Day, Month, Year, etc. These attributes are necessary for scheduling and synchronizing.
  - *Interval:* The Interval class represents a specific period of time and is related to the Time class. Each Interval holds a "startTime" and an "endTime" object property (Interval → domain, Time → range). Every Interval instance may also be related to other Interval instances via various object properties such as: before, after, during, overlapping, etc.

- *Environment Domain Ontology*
  - *EnvironmentalProfile*: The various environmental parameters are represented by the EnvironmentalProfile class. It incorporates various datatype properties such as: Temperature, Humidity, Pressure, WindSpeed, Visibility, Noise, Illumination, etc.

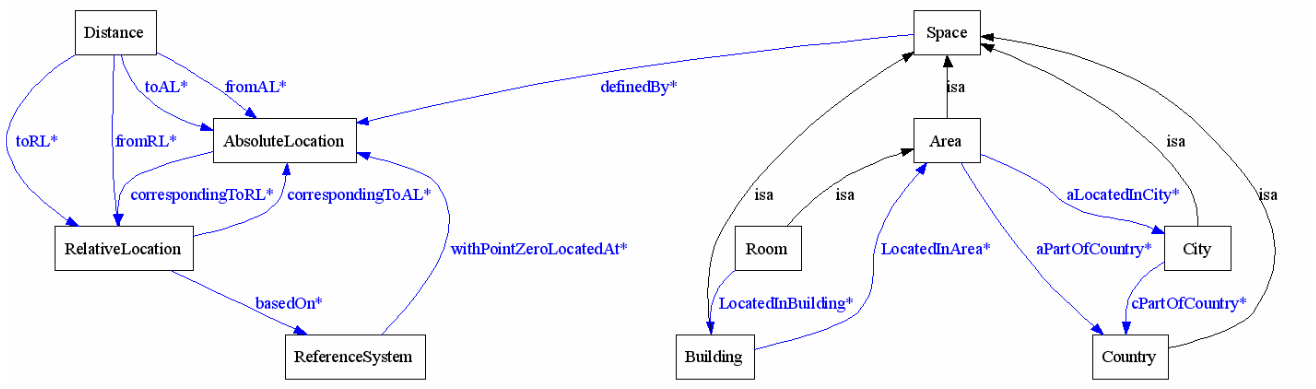

Figure 5 The Spatial domain context ontology

2.5.1.5 The Object domain context ontology

We introduce here the object domain context ontology. The central class of this domain is the Object class. It subsumes any material object, i.e. any object that has a substance. In the Amigo home application domain, this class will be used to represent chairs, tables, books. It should be the job of application developers to specialize the Object class for example by creating a "Furniture" subclass, which might be further specialized into a "Chair" subclass. Real chairs will then be represented as instances of this "Chair" subclass.

At the level of this abstract Object class we define properties that are shared by any material object. They include:

- Its location. The same schema will be used as for defining user location. More specifically, the Object class will have a "hasAbsoluteLocation" object property which range is the AbsoluteLocation class.
- Its weight. This will be represented as a datatype property.
- Its owner. This will be represented by an object property which range is the User class.
- Its shape. Depending on the Amigo applications needs, a specific ontology might be needed for modeling geometrical shapes. This ontology could be used here for relating an object to its shape as well as in relation to the spatial domain ontology introduced earlier if shapes of places such as buildings, rooms are needed.

It is worth noticing that the Object class shares relationship with classes from other domains, for instance with the User class through the "ownsBy" property, with the Physical domain while representing objects locations, even the Device domain if some device is to be controlled by manipulating a simple object such as a cup, that would be used as a tangible interface to the service provided by this device, or if the device lies on top of a table. Potential alignment with existing classes will be studied for factorizing shared property. For example it might be relevant to create a superclass of the Object class for holding the weight datatype property which has already been defined at the level of the User class.

We then should view the context ontology as a flexible and evolving model which should provide support for specialization when handed over to application developers.

2.5.2 QoS

As stated in [Amigo-D2.1], the Quality of Service (QoS) can be seen as a collection of means offering guarantees to users with respect to the applications. QoS properties are critical information necessary for the dynamic selection of the most appropriate services with regards to the user requirements and objectives. In case various services have been discovered that meet the functional service requirements of the user, then considering the QoS parameters enables the Amigo middleware to rank these services and identify the one that best meets the user preferences. In addition to the service selection issue, the QoS aspect is also important in the process of composing various services with different attributes and requirements.

The discussion on QoS aspects and related ontologies has been started in scope of WP2 [Amigo-D2.1]. Some initial categories that can be important from Amigo point of view have been identified. These are (i) runtime-related QoS (ii) transaction-support QoS, (iii) configuration- & cost-related QoS and (iv) security-related QoS. Here we continue our work, analysing Amigo scenarios and existing QoS related vocabulary ontologies and classification to achieve flexible and integrated QoS schemes in AmI environments. Our objective is to built the QoS ontology in two complementary levels. The upper level provides a generic QoS attribute model and its basic features for the language ontologies (discussed in D3.1b), whereas the lower level provides a QoS class hierarchy for the vocabulary ontologies, with a full description of the concrete quality parameters along with the relationships between them (this document, D3.1a).

2.5.2.1 Overview of available classifications and QoS related ontologies

In this section an overview of most important recent research work on the design of QoS ontologies is provided.

In [ZhCL04] the defined QoS ontology consists of three layers: the QoS profile layer, which is designed in order to provide matchmaking functionality; the QoS property definition layer, which is used for elaborating the property's domain and range constraints; and the metrics layer that provides metrics definition and measurement functionality. A drawback of this modelling approach is that it is specifically designed for matchmaking purposes and thus it does not provide a generic QoS ontology language. Furthermore, a QoS ontology vocabulary is absent.

The framework presented in [MaSi04] is based on agents, which are used for the dynamic selection of web services. Service providers publish their services to registries and agencies, and service consumers use their agents in order to discover the desired service. A QoS ontology language is designed that includes the basic characteristics of various quality parameters and the main concepts associated with them. Additionally, a QoS ontology vocabulary is presented that specifies the domain-independent quality concepts and is typically completed by a domain-specific lower ontology. A disadvantage in this work is that the metrics concept is absent.

In [TGNR03] a web service QoS XML Schema is defined. [TGNR03] classifies the QoS parameters in two main categories: network related parameters and server-client related parameters. In order to establish a matchmaking mechanism, this approach extends the traditional web service architecture of publishing and requesting a service through the UDDI registry with the concept of a service broker. An advantage of this approach is that it enables users to monitor the status of the server. Nevertheless, it does not provide an advanced QoS ontology, but only a simple XML Schema.

Another interesting work is presented in [SHU03]. This approach also focuses on extending the current UDDI Model in order to integrate QoS-based functionality. Even though it does not provide a QoS ontology language or vocabulary, nevertheless the enumeration of QoS parameters combined with the provided classification are interesting and useful in various domains.

Concluding, the [W3C] Consortium provides a quite detailed enumeration of Web Services related QoS parameters. Possible approaches for having Web Services QoS support are investigated based on the extension of current UDDI and SOAP models. [W3C] has not published yet an official QoS ontology. Nevertheless, the extended list of QoS parameters it provides can be used for the design of a QoS ontology vocabulary.

2.5.2.2 Analysis of QoS classifications

In this section an instantiation of the designed QoS ontology is described for the AmI home domain. The fragment from the Amigo 'Extended home' scenario [Amigo-D1.1] may provide an instantiation of the proposed QoS classifications:

…Roberto and his grandfather John have continued their habit of playing games together, watching a bit of TV and having their man-to-man chat. Amigo selects the games that both John and Roberto like. They can look at each other and see what game moves are being made. Amigo can also set-up a video-conference for them in which they can watch TV together, show the newest acquisitions of their collections, or just tell their stories...

Subsequently we will refer to the service above as "AmbientPresence". The aforementioned scenario incorporates a variety of QoS parameters that should be satisfied in order to ensure an effective and reliable communication from each user's view. The most important QoS parameters in the specific scenario are those related to network performance, as the scenario involves real-time video communication. Thus, Jitter, ResponseTime and Latency should be kept low, high Throughput should be guaranteed, while the ErrorRate quality restriction is looser. Indicative values for these parameters are given below. The "AmbientPresence" service should be Accessible and Available as long as possible, satisfying user's needs of keeping in touch as if they were on the same house. In addition, Scalability is also important as the server should be capable of providing additional resources whenever required, while its Capacity should lie above a specific threshold. The Cost QoS parameter of this service expresses how much the service provider charges the user and what charging scheme it uses (e.g. fixed cost per month). Another essential parameter is SupportedStandard that indicates the various standards supported by the service. Finally, in order to have the users as satisfied as possible, high Reliability should also be ensured.

Other parameters as Stability, Regulatory, Accuracy and Integrity are not considered in this particular example, but may be involved in other situations and scenarios e.g. Home care and safety. Also Security parameters will be further analysed in our future work.

The initial taxonomy for QoS domain and the description of QoS parameter classes is presented below.

QoSParameter

- *Accessibility*
- *Accuracy*
- *Availability*
- *Capacity*
- *Configuration*
  - *Regulatory*
  - *Stability*
  - *SupportedStandard*
- *Cost*
- *Integrity*
- *Performance*
  - *ErrorRate*

Jitter

Latency

ResponseTime

Throughput

Reliability

MTBF

MTTR

Scalability

Security

Auditability

Authentication

Authorization

Confidentiality

DataEncryption

NonRepudiation

Performance

The Performance QoS parameter aggregates information that mainly depends on the properties of the network connection between the user and the service provider. It represents how fast a service request can be completed. The subclasses of the Performance class in our QoS ontology are described below.

- *Throughput*. It represents the number of service requests served at a given period of time. Usually users are interested in high throughput, especially when they need to send/receive large volumes of data. When real time applications are running on the user side, high throughput is absolutely necessary. QoS measures can include the maximum throughput or a function that describes how throughput varies with load intensity. It is a dynamic parameter that depends on the network properties and state at the time measured. High throughput can be the result of light traffic, whereas in cases where the load is heavy, throughput may be unendurably low. The main issue in monitoring throughput is when and where to measure it, in order to obtain measurements that represent the actual situation.
- *Latency*. It refers to the time required so that a packet is transferred from the user node to the service provider node and vice versa. For the majority of services, latency should not exceed some specified thresholds. Like throughput, the user-provider and the provider-user latencies may differ. The round-trip time is the sum of the above latencies and it may also be of interest. For instance round-trip time is an overhead which applies to all request-response communications. Latency also varies over time and thus, it shares the same problems with throughput concerning how to obtain a representative measurement.
- *ResponseTime*. It is defined as the total time needed by the service requester to invoke the service. It is measured as the time interval between the moment when the requester initiates the invocation and the moment when the requester receives the last byte of the response. Response time sometimes refers to the guaranteed time required for the completion of the request. This expression of the ResponseTime might involve the minimum time or the maximum, depending on the purpose of its measurement.

- *Jitter*: It refers to the variation (standard deviation) of the latency. Jitter increases as network load increases. For instance, severe jitter can occur when packets travel through different routing paths because of routers being congested. Jitter is also different in the two communication directions.
- *ErrorRate*. It expresses the frequency of packet losses. It is a dynamic real time parameter that depends on the lower layer network protocols used, the node properties and the network load.

In general, high performance services should provide high throughput, low latency, fast response time, low jitter and low error rate.

Scalability

The Scalability QoS parameter is used to describe the server's ability to increase its computing capacity and the system's ability to process more users' requests, operations or transactions in a given time interval. It is evident that the more scalable a server is the more efficient it is in handling varying numbers of requests. Scalability is related to performance.

Availability

Availability is defined as the probability that the server is up and running in order to handle incoming requests. Availability varies between 0 and 1. When it is closer to 1, the server is considered to be highly available, a feature always necessary.

Accessibility

Accessibility refers to server's ability of serving a web service request. Accessibility is different from availability, as a service maybe available, which means that the server is up, but can not handle any incoming requests, thus not being accessible. This situation frequently occurs when server's system has a lot of incoming requests and is not scalable. High accessibility can be achieved, e.g., by building highly scalable systems.

Accuracy

This QoS parameter refers to the accuracy of results in a numerical manner. It specifies the number of significant decimal digits of results. For example, time can be measured with accuracy of 1, 2 or more decimal digits.

Capacity

Capacity specifies the maximum number of concurrent requests a server is able to handle simultaneously. The more capacity a server has, the more requests it can handle. Thus, the higher the server capacity is, the higher its availability and accessibility are.

Cost

This QoS parameter represents the overall cost that results from service usage. The Cost parameter is critical for the service selection, as usually the service cost is identified by the user as one of the most significant QoS parameters. The service cost may be expressed in various ways, e.g. cost per request, per data volume, per time, etc.

Configuration

The configuration of services is related to the interface update procedure and/or the adopted standards; and it provides information about the regulations the service complies with. It indicates whether the services can interact with each other. Configuration is measured by the following metrics:

- *Stability*. It represents how often a service is modified with respect to the service interfaces. This parameter also refers to the changes in the service implementation.
- *SupportedStandard*. It is the parameter which refers to the standards that the service complies with. Web services use plenty of standards such as UDDI, WSDL, SOAP, etc. The standards that a service complies with make the service more or less interoperable with the other services.
- *Regulatory*. The Regulatory parameter refers to the probability of the fact that the service is compliant with a random regulation.

Integrity

Integrity represents the ability of a web service to preserve data integrity during a transaction. In order to accomplish data integrity, all the transactions that implement a specified function are handled as a single unit. In case the transaction is completed successfully all changes in data are committed, while if the transaction is not completed, all changes are rolled back. Integrity of data is a boolean QoS parameter that is either supported by the service or not.

Reliability

This QoS parameter represents the possibility of a service to be completed successfully. Reliability is closely related to availability, as the more available a server is, the more reliable its services are. The parameters that measure reliability are:

- *MTTR* (Mean Time To Repair). MTTR is the average time that is required for the server to start functioning properly again after a failure.
- *MTBF* (Mean Time Between Failures). MTBF is the average time between two subsequent failures of the service server.

Security

This category of QoS parameters refers to the security level a service provides[1]. Security is of great importance in the Amigo environment, as one of its objectives is to keep the user in control, avoid the cognitive saturation of the user and respect the user privacy. The security subclasses in the designed QoS ontology are presented below.

- *Confidentiality*. It refers to the way the service treats data so that only authorized parties can access or modify the data. This parameter ensures that secured data are not gathered to places where authorization mechanisms are not in place.
- *Auditability*. Auditability ensures that the service history of invocation and execution can be checked and reviewed. It belongs to the same category as Traceability according to many approaches, which is here incorporated to the Auditability parameter for convenience.
- *Authentication*. It is one of the most important issues in the process of service invocation and usage. It takes place during the initial steps after the service invocation request, when the service provider asks the user to prove his/her identity. If the authentication is successful, the user is able to start using the requested service. The exact authentication mechanism (e.g. password verification, physical token recognition, biometric methods, etc) is represented in our QoS ontology as the string equivalent of its widely known name.

[1] It has been argued that the security parameters are functional properties and thus, they should not be integrated in the QoS ontology. Nevertheless, until a final decision is reached and since in most QoS ontologies in the literature various security parameters are included, we chose not to remove the security branch from the QoS ontology vocabulary for the time being.

- *Authorization*. Authorization is in general the process of verifying that the user is permitted to perform a specific action. In the QoS ontology vocabulary it refers to the procedure that decides whether the user is allowed or not to temporarily access the specified services after he/she has been authenticated by the system. Authorization mechanisms are used to validate the access rights to protected services. In order for users to get authorised, they must have the appropriate credentials.
- *Data Encryption*. Refers to the way the service encrypts the data, i.e. the algorithms used. It is different from the two parameters above as it covers data transfer when the service is in use. Data encryption is initiated by the user who submits a request to the encrypted data provider that replies whether it can support the specified scheme or not.
- *Non Repudiation*. Systems must ensure that a party cannot repudiate (reject) a transaction after its completion. To protect and ensure digital trust, parties in such systems may employ Digital Signatures, which will not only validate the sender, but will also 'time stamp' the transaction, so it cannot be claimed subsequently that the transaction was not authorized or is not valid for any reason.

Note here that the maximum height of the relevant taxonomy tree is two. This is preferable for a reduced complexity development. This taxonomy will be further refined and analysed against the Amigo aware service discovery requirements. Also the relations with context information will be carefully validated as some QoS may depend on context information and in some cases context can be viewed as one of the constraints for the successful service discovery and selection with the required QoS.

3 Methods and tools for semantic modelling of services

As discussed earlier, the availability of development tools that are robust, simple and easy to use is a quite critical issue in Amigo. Amigo is planning to use an OWL based language for semantic modelling, and open source approach wherever possible. There are numerous commercial and open source tools available for semantic modelling. An optional approach is to use Unified Modeling Language (UML) that is a widely used technique for modelling in software engineering and supports integration of various tools, even as open source.

A survey of the existing tools on semantic service modelling supporting these is presented in the Table 1. After evaluating the surveyed tools, Protégé has been selected to be used for modelling the Amigo domain taxonomies. In spite of opposite claims, this tool proved to be easy to learn and adopt, at least for simple taxonomy modelling, visualising, and exporting it to OWL.

Table 1 Semantic modelling tools

Tool Name	Description
Protégé	Protégé is platform independent open source ontology editor. Protégé is based on Java and it is extensible by plug-ins. There are many plug-ins that enable visual editing of ontologies, for example ezOWL, OntoViz and OWLViz. There is also a plug-in that enables export and import in UML version 1.4 XML format. Protégé supports Frames, XML Schema, RDF(S) and the Web Ontology Language (OWL). Protégé is available as free software under the open-source Mozilla Public License. http://protege.stanford.edu/
SWOOP	Open source OWL Ontology editor that employs web-browser metaphor for its design and usage. SWOOP has plug-in architecture for new presentation syntax tabs and it uses Manchester (WonderWeb) OWL API. SWOOP is released under GNU Lesser General Public License (LGPL). http://www.mindswap.org/2004/SWOOP/
The OWL-S Editor	Open source editor for creating OWL-S services. It is Tab Widget plug-in to Protégé Ontology Editor. Provides editing modes for four classes of OWL-S service: Service, Profile, Process and Grounding. The source code is distributed under Mozilla Public License version 1.1. http://owlseditor.semwebcentral.org/
OWL-S Editor	Java-based tool for creating, validating and visualizing OWL-S models. OWL-S descriptions can be made from templates or through “OwlsWiz” wizard which generates basic OWL-S description from WSDL input (file). At the heart of OwlsWiz is “Visual Composer” that allows composing of atomic processes using UML Activity Diagrams. OWL-S editor uses the Java based Jena API for validating the syntax and integrity of ontology. http://staff.um.edu.mt/cabe2/supervising/undergraduate/owlseditFYP/OwlSEdit.html
ezOWL	Supports much of the OWL Full ontology model. Enables UML-like diagrammatic definitions. Currently ezOWL is available as plug-in to Protégé

	tool. Standalone version is under development.
OntoTrack	Graphical editing of classes in UML-style with inference feedback. OntoTrack is pure Java application that supports OWL Lite model. OntoTrack requires the external Racer ontology reasoner in order to provide instant feedback about relevant modelling consequences. http://www.informatik.uni-ulm.de/ki/OntoTrack/
OntoEdit / OntoStudio	OntoEdit is an Ontology Engineering Environment supporting the development and maintenance of ontologies using graphical means. OntoEdit (version 0.6) supports F-Logic, RDF-Schema and OIL. OntoEdit has successor called OntoStudio that is commercial product but can be downloaded for three months free evaluation period. OntoStudio supports OWL and RDF. OntoStudio is based on IBM's development environment Eclipse. http://www.ontoknowledge.org/tools/ontoedit.shtml http://www.ontoprise.de/content/e3/e43/index_eng.html
WebODE	WebODE is Java based tool that has been built using 3-tier architecture. It supports many formats (both export and import) including UML, OWL, RDF(S). There is no downloadable version of WebODE, only web access to the service that requires login/password. http://webode.dia.fi.upm.es/WebODEWeb/index.html
Cerebra Construct	Visual modelling environment for ontologies. Integrated with Microsoft's Visio environment and Cerebra Server. Supports OWL-DL. Commercial software. http://cerebra.com/
IODE (Integrated Ontology Development Environment)	Commercial tool for ontology development. Supports OWL and RDF. http://www.ontologyworks.com/
SemTalk	SemTalk is add-on for MS Visio for modelling ontologies and processes. SemTalk supports RDF, DAML and OWL. SemTalk is commercial product. http://www.semtalk.com/
Unicorn Workbench	Commercial Java based tool for graphical ontology modelling. Supports XML Schema, RDFS, OWL, XMI (XML Metadata Interchange). http://www.unicorn.com
SMORE	Allows creating OWL markup for HTML Web pages. http://www.mindswap.org/2005/SMORE/

Some authors have proposed that UML should be used in ontology modelling because current ontology editor tools may be too complex for the majority of software developers, and thus, a UML modelling tool they are already familiar with would be easier to adopt. A problem of using UML based tools in Amigo may be that UML does not directly support all the required OWL constructs. Also the XML export formats used by current UML tools, may greatly vary from one tool to another. Object Management Group is currently working towards the design of a solution addressing this situation [ODM03]. In their proposal they are looking for a solution that would enable UML models to be converted into ontology models (OWL-DL). The following diagram (Figure 6) illustrates this approach.

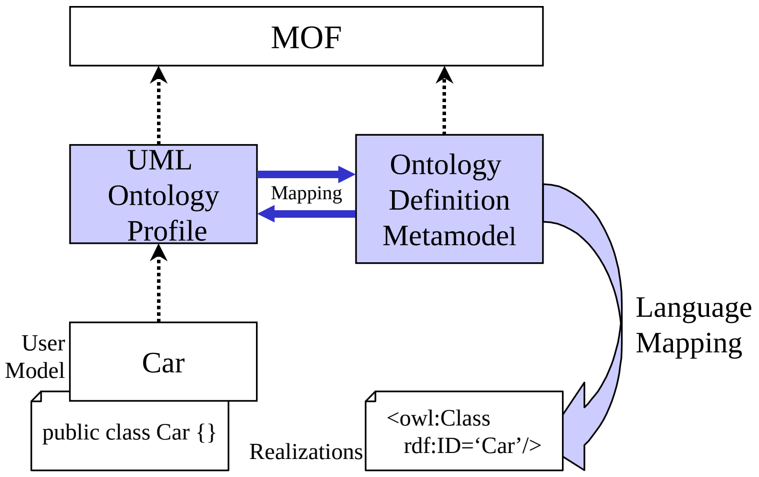

Figure 6 Object Management Group proposal for UML ontology modelling.

In conclusion, it should be as easy as possible for modelers to design ontologies and Protégé as a preliminary choice seems to be the most promising one for the Amigo needs. In the future, the ontology UML Profile approach could be considered as well, as UML is well known and widely supported by the research and development community. Many ontology modelling tools offer support for OWL. However, they may introduce an additional learning step for a software designer. The integration of ontology based design tools and UML based tools in a single tool might be very useful and desirable, as it could facilitate and speed up the adoption of a semantic modelling approach.

4 Conclusions

The objective of Amigo project is to effectively integrate and compose diverse heterogeneous services available in our homes from different application domains, i.e. CE, mobile, PC, and domotic domains. The semantic modelling of Amigo services based on language- and vocabulary ontologies will enable interoperability of heterogeneous services. This document started presenting an analysis on the kind of vocabularies that are necessary in different Amigo domains to cope with heterogeneity of service descriptions. A step-by-step analysis (outlined in the introduction of Chapter 2) follows with regards to the Amigo scenarios and user requirements, surveying numerous information sources, and discussing existent classifications and ontologies in the domains of interest. The devices and their functional properties have been initially studied and analysed. Also non-functional device aspects required to adapt the Amigo services to particular devices and to select a device based on the capabilities it provides, are identified in the class hierarchies. In a user centric and dynamic environment such as the home, service discovery, selection and composition should be adaptive also according to available resources at specific time and place, considering the current user activity and status situation, and the user preferences. Special attention has been paid to this kind of context information. The initial classifications, analysis and discussions on different levels of ontologies required for Amigo specific domains (i.e. domotic, CE, mobile, PC) including discussions on non-function aspects of the service have been presented. In our future work these initial classifications and vocabularies will be further verified, extended and validated against service/application developer’s and user’s requirements for efficient service discovery, invocation and service composition. We will also aim to integrate and extend the existing ontologies and provide recommendations for service developers on how to use them.

As already stated, semantic modelling of functional and non-functional properties of Amigo services will enable dynamic ‘ad hoc’ service composition, adaptive to information related to user and services. Adaptive ‘ad hoc’ service composition will provide the necessary means to help users to integrate and configure devices and services available at home in a transparent and useful manner to achieve safer, secure, and more comfortable environment at home. There are plenty of examples that can be found from our today’s home and also inspirited by extensive user research performed in scope of the Amigo project [Amigo-D1.1], to demonstrate the necessity of adaptive service composition. Some of them are realized in Amigo scenarios: Amigo adapts the home environment to the video game by configuring lights, sounds and video throughout the room; Amigo can also set-up a video conference for John and Robert in which they can watch TV together or play ‘hide-and-seek’ with little Pablo; when the washing machine finishes its work, Amigo as a housekeeper, notifies Maria on TV or by switching on/off the lamp in the kitchen, depends on Maria’s location and activity; as a doorkeeper Amigo is able to recognize a visitor, notify residents by some means (e.g. voice or message on TV) and open the door.

In the scenes above different simple services and technologies are integrated in complex composite services. Lighting, audio, and video services create ambient gaming atmosphere. Lighting, messaging, and TV systems notify about washing machine status. Combination of IP phone, video camera and TV enables enjoyable video conferencing service. Video camera or voice recognition system, door system, possibly TV and messaging services are composed by Amigo doorkeeper.

As stated [Amigo-D2.1], service composition needs to be adapted according to environment’s specifics and user preferences. Adaptation of composite services is defined by ad-hoc scenarios that specify the sequence of actions that must be performed based on user preferences and on the available set of devices. To support such scenarios in an Amigo system, various approaches are possible. However, the research in this direction is in its early stages and candidate solutions must be studied and assessed.

The adaptive composition process is illustrated in Figure 7. Events that require adaptation are for example events that inform about the context or QoS change, service user specific actions, or service discovery related events. The responsibility of adaptation mechanisms is to identify and compose the optimal adaptation to current situation using the set of capabilities provided by the services available.

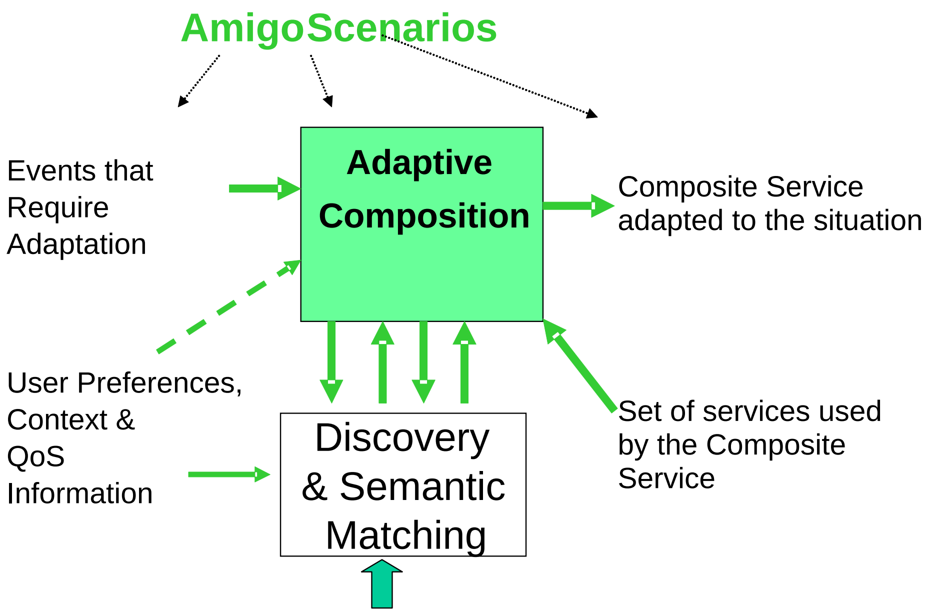
AmigoScenarios
Adaptive Composition
Events that Require Adaptation
Composite Service adapted to the situation
User Preferences, Context & QoS Information
Discovery & Semantic Matching
Set of services used by the Composite Service

Figure 7 The schematic for adaptive service composition

One approach consists of describing a functional composite service in the form of an abstract workflow that can be executed by integrating on the fly services that are available in the home environment. The mapping of abstract workflow to the concrete set of services is achieved by performing semantic matching of services interfaces (i.e. inputs, outputs), thus selecting candidate services for the following integration and conversation matching on the set of selected services. However, this functional service composition framework should be further complemented by integrating of non-functional attributes of service (i.e. QoS), context and user preferences and profiles to facilitate composite abstract workflows specification.

Another approach relies on introducing a programming language dedicated to the domain of ubiquitous computing. This language enables a user to express the rules that describe the sequence of actions to perform when an event occurs. Although expressive, a language approach may fail to be accessible to non-programmers. We propose to design and develop a language dedicated to the domain of service composition to describe scenarios of ambient intelligence. Domain-specific languages (DSLs) have been successfully used in various application areas and have shown their benefits in term of accessibility to domain experts, conciseness, readability, safety and robustness. The DSL targeting service composition will offer dedicated abstractions and notations to the domain experts that rely on the ontology defined in Task 3.1. A graphical user interface can be provided to the user to construct such a scenario by dragging and dropping built-in visual components.

In addition to these approaches, various other software engineering approaches are currently investigated. Design patterns provide well known solutions to handle complexity associated with software architectures, and complement the service oriented approach used in Amigo. An example of the use of design patterns can be found in adaptive object models [JYRJ02]. Event Condition Action paradigm (ECA) is an approach for handling adaptation for dynamic environment. Feature modelling is associated to many domain analysis methods and is used in generative programming and product line software architectures. Feature models [KA90][KE00] can provide the semantic model for describing variability related to adaptation, i.e. choices that can be made and their mutual relations, complementing the use of domain vocabularies described in this document.

Acronyms

ACAI	Agent-based Context-Aware Infrastructure
ADPCM	Adaptive Differential Pulse Code Modulation
ASC	Aspect Scale Context
AVI	Audio Video Interleave
CoBrA	Context Broker Architecture
CC/PP	Composite Capability/Preference Profiles
CD	Compact Disc
CE	Consumer Electronics
CoOL	Context Ontology Language
DAML	DARPA Agent Markup Language
DIVX	Digital Video express
DSL	Domain Specific Language
DVD	Digital Versatile Disc
ECA	Event Condition Action paradigm
EDGE	Enhanced Data GSM Environment
EPG	Electronic Program Guide
FIPA	Foundation for Intelligent User Agents
JVM	Java Virtual Machine
GIF	Graphics Interchange Format
HTML	HyperText Markup Language
JPEG	Joint Photographic Experts Group
LGPL	Lesser General Public Licence
MIDI	Musical Instrument Digital Interface
MMS	Multimedia Message Service
MPEG	Moving Picture Experts Group
MUPE	Multi-User Publishing Environment
ODM	Ontology Definition Metamodel
OGG	Ogg Vorbis audio compression format
OMA	Open Mobile Alliance
OUP	Ontology UML Profile
OWL	Web Ontology Language
STB	Set-Top Box
PC	Personal Computing
PCM	Pulse Code Modulation
PDA	Personal Digital Assistant
QoC	Quality of Context

QoD	Quality of Device
QoS	Quality of Service
RDF	Resource Description Framework
RTF	Rich Text Format
SMS	Short Message Service
TIFF	Tagged Image File Format
XML	Extensible Markup Language
UAProf	User Agent Profile
UI	User Interface
UML	Unified Modeling Language
USB	Universal Serial Bus
VoIP	Voice over Internet Protocol
WAP	Wireless Application Protocol
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network

References

[Amigo-D1.1] Deliverable D1.1 User research results, January 2005

[Amigo-D2.1] Amigo Consortium. Deliverable D2.1: Specification of the Amigo Abstract Middleware Architecture. April 2005.

[Amigo-D3.1b] Amigo Consortium. Deliverable D3.1b: Detailed Design of the Amigo Middleware Core; Service Specification, Interoperable Middleware Core, September 2005.

[BPRC04] Bandara, Payne, De Roure and Clemo,.An Ontological Framework for Semantic Description of Devices, 2004.

[BuKS03] T. Buchholz, A. Küpper, M. Schiffers, Quality of Context: What is it and why we need it, Workshop of the HP OpenView University Association, Geneva, Switzerland, July 2003.

[CCPP] Composite Capabilities / Preference Profiles http://www.w3.org/Mobile/CCPP/

[ChFi03] H. Chen, T. Finin, An Ontology for Context Aware Pervasive Computing Environments, *Workshop on Ontologies and Distributed Systems 2003*, Acapulco, Mexico, August 2003.

[DeSA01] A.K. Dey, D. Salber, G.D. Abowd, A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, *Human-Computer Interaction Journal,* Vol. 16, No. 2–4, pp. 97–166, 2001.

[DIP03] Device Independence Principles. http://www.w3.org/TR/2003/NOTE-di-princ-20030901/

[FIPA79] FIPA FIPA Agent Software Integration Specification http://www.fipa.org/specs/fipa00079/

[FIPA91] FIPA Device Ontology Specification. 2001. http://www.fipa.org/specs/fipa00091/PC00091A.html

[HuVo04] Bodo Hüsemann, Gottfried Vossen, Ontology-Driven Multimedia Object Management. European Research Center for Information Systems University of Muenster, Germany, 2004. http://www.sigsemis.org/articles/husemann/document_view

[FIPAAV] FIPA Audio/Visual Entertainment and Broadcasting Specification http://www.fipa.org/specs/fipa00081/XC00081B.html#_Toc505481810

[JYRJ02] Joseph W. Yoder , Ralph E. Johnson, The Adaptive Object-Model Architectural Style, *Proceedings of the IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture: System Design, Development and Maintenance*, p.3-27, August 25-30, 2002

[GuPZ04] T. Gu, H.K. Pung, D.Q. Zhang, Toward an OSGi-Based Infrastructure for Context-Aware Applications, *IEEE Pervasive Computing Magazine*, Vol. 3, No. 4, pp. 66-74, October, 2004.

[KA90]	Kang, K. et. al: Feature-Oriented Domain Analysis (FODA) Feasibility Study, Technical Report No. CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsbulrgh, 1990. Pennsylvania
[KE00]	Krzysztof Czarnecki and Ulrich W. Eisenecker, Generative Programming, Addison-Wesley, 2000, ISBN 0-201-30977-7
[KhKa05]	M. Khedr, A. Karmouch, ACAI: Agent-Based Context-aware Infrastructure for Spontaneous Applications, *Journal of Network & Computer Applications*, Vol. 28, No. 1, pp. 19-44, May, 2005.
[MaSi04]	E. Michael Maximilien, Munindar P. Singh, A Framework and Ontology for Dynamic Web Services Selection, Vol. 8, No. 5, pp. 84-93, September/October, 2004.
[ODM03]	Ontology Definition Metamodel Request For Proposal. Object Management Group http://www.omg.org/docs/ad/03-03-40.pdf
[OMA05]	OMA User Agent http://www.openmobilealliance.org/release_program/uap_v20.html
[OWL]	OWL semantics and abstract syntax http://www.w3.org/TR/owl-semantics/
[PBWG04]	D. Preuveneers, J.V. Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, K.D. Bosschere, Towards an extensible context ontology for Ambient Intelligence, *European Symposium on Ambient Intelligence*, Eindhoven, Netherlands, November, 2004.
[StLF03]	T. Strang, C. Linnhoff–Popien, K. Frank, CoOL: A Context Ontology Language to enable Contextual Interoperability, *4th IFIP International Conferece on Distributed Applications & Interoperable Systems*, Paris, France, November, 2003.
[SHU03]	Shuping Ran, A Model for Web Services Discovery With QoS, *ACM SIGecom Exchanges*, Vol. 4, No. 1, March, 2003.
[SKWIH]	Shimizu, Kitagawa, Wan, Ishikawa, Hjelm. *A Device and Service Description Framework for Discovering and Reasoning in Autonomous P2P Environment.*
[Swoogle]	http://swoogle.umbc.edu/
[TGNR03]	M. Tian, A. Gramm, T. Naumowicz, H. Ritter, J. Schiller, A Concept for QoS Integration in Web Services, *1st Web Services Quality Workshop*, Rome, Italy, 2003.
[TSPO04]	Tsinaraki, Polydoros, Kazasis & Christodoulakis, *Coupling OWL with MPEG-7 and TV-Anytime for Domain-specific Multimedia Information Integration and Retrieva,* 2004. http://www.ced.tuc.gr/Staff/Director/Publications/publ_files/C_TPMC_RIAO_2004.pdf
[TvAnytime]	TV-Anytime Forum. http://www.tv-anytime.org/

[TUT]	Ontology development tutorial http://protege.stanford.edu/publications/ontology_development/ontology101.html
[UAProf]	User Agent Profile Specification. Wireless Application Protocol Forum Ltd., 1999. http://www.wapforum.org/
[WZGP03]	X. Wang, D. Zhang, T. Gu, H. Pung, Ontology Based Context Modeling and Reasoning using OWL, *Workshop on Context Modeling and Reasoning at IEEE International Conference on Pervasive Computing and Communication*, pp. 18-22, March, 2004.
[W3C]	K.Lee, J.Jeon, W.Lee, S. Jeong, QoS for Web Services: Requirements and Possible Approaches, W3C Working Group Note, November, 2003.
[ZhCL04]	Chen Zhou, Liang-Tien Chia, Bu-Sung Lee, DAML-QoS Ontology for Web Services, *International Conference on Web Service*, pp. 472-479, 2004.