



Contract No. IST 2005-034891

Hydra

**Networked Embedded System middleware for
Heterogeneous physical devices in a distributed architecture**

D2.2 Initial Technology Watch Report

**Integrated Project
SO 2.5.3 Embedded systems**

Project start date: 1st July 2006

Duration: 48 months

**Published by the Hydra Consortium
Coordinating Partner: C International Ltd.**

2 February 2007 - version 1.6

**Project co-funded by the European Commission
within the Sixth Framework Programme (2002 - 2006)**

Dissemination Level: Confidential

Document file: D2.2 Initial technology watch report.doc

Work package: WP2 – Iterative User Requirements Engineering

Task: T2.2 – Initial requirements specifications

Document owner: TUK

Document history:

Version	Author(s)	Date	Changes made
1.0	Peter Kostelnik (TUK), Martin Sarnovsky (TUK)	26-09-2006	
1.1	Tomas Sabol (TUK)	02-10-2006	Minor changes in structuring, the content
1.2	Peter Kostelnik (TUK), Heinz-Josef Eikerling (C-LAB)	16-10-2006	Added section on wireless devices
1.3	Martin Sarnovsky (TUK), Peter Rosengren (CNET), Pablo Antolín Rafael (TID), Stephan Engberg (PRIWAY), Klaus Marius Hansen (UAAR), Kristian Ellebæk Kjær (UAAR), Peter Kostelnik (TUK)	05-11-2006	Refined sections: Embedded AmI, Wireless Nets and Devices (TID), MDA (CNET); Added section on privacy and security
1.4	Peter Kostelnik (TUK), Dasa Lackova (TUK), Tomas Sabol (TUK)	07-11-2006	Added section: Conclusions; Final editing
1.5	Siegfried Bublitz (C-LAB), Dasa Lackova (TUK)	11-12-2006	General corrections; Final editing
1.6	Peter Kostelnik	02-02-2007	Refined sections on SOA and Ontology-based knowledge modelling, general corrections, final editing
1.6	Jesper Thestrup (IN-JET)		Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Comments
Jesper Thestrup (IN-JET)	17-12-2006	Minor suggestions, conclusions
Mario Kupries (UAAR)	25-01-2007	Vocabulary and references refined, references to standards, SOA and MDA clarifications requested.

Index:

1	Introduction	6
2	Executive summary	7
3	Embedded ambient intelligence	9
3.1	Introduction	9
3.2	Context-aware middleware	10
3.2.1	Aura	11
3.2.2	CARMEN.....	11
3.2.3	CARISMA	11
3.2.4	CoBrA (Context Broker Architecture)	12
3.2.5	Context Toolkit	12
3.2.6	Cooltown.....	12
3.2.7	CORTEX	13
3.2.8	Gaia	13
3.2.9	Hydrogen Context-Framework	13
3.2.10	Middlewhere.....	14
3.2.11	MobiPADS	14
3.2.12	SOCAM	15
3.2.13	Stick-e Notes.....	15
3.3	Emerging opportunities	15
3.3.1	Interactive devices	15
3.3.2	Interaction paradigm	16
3.3.3	Interaction distribution	16
3.3.4	Automation versus human control	17
3.3.5	Content and functionality	17
3.3.6	Health and safety	17
3.3.7	Privacy and security	18
3.4	Conclusion	18
3.5	References.....	18
4	Semantic web	21
4.1	Formalisms for modelling web services.....	23
4.1.1	OWL-S (Web Ontology Language for Services).....	23
4.1.2	WSMO (Web Service Modelling Ontology).....	24
4.1.3	WSDL-S (Web Service Semantics).....	24
4.1.4	BP4WS (Business Process Execution Language for Web Services)	25
4.2	Frameworks and tools for semantic web services	25
4.2.1	OWL-S Tools.....	25
4.2.2	WSMX (Web Service Execution Environment)	26
4.2.3	IRS (Internet Reasoning Service) III.....	27
4.2.4	METEOR-S.....	27
4.3	References.....	27
5	Ontology-based knowledge modelling	29
5.1	Ontology.....	29
5.1.1	Ontology languages.....	29
5.1.2	Ontology evolution	34
5.1.3	Ontology versioning.....	35
5.2	Change management	36
5.2.1	Content syndication.....	36
5.2.2	Version control	40
5.2.3	Ontology-based change management	42
5.3	References.....	42
6	Service-oriented architecture	45
6.1	Key elements of SOA.....	45

6.1.1	Service	45
6.1.2	Messages	46
6.1.3	Dynamic discovery	46
6.2	Service-orientation principles	46
6.2.1	Service reusability	47
6.2.2	Service contract	47
6.2.3	Service loose-coupling	47
6.2.4	Service abstraction	47
6.2.5	Service composability	48
6.2.6	Service autonomy	48
6.2.7	Service statelessness	48
6.2.8	Service discoverability	49
6.3	Web services	49
6.3.1	Messaging specifications	50
6.3.2	Service description	54
6.3.3	Discovery specifications	56
6.3.4	Security specifications	58
6.3.5	Business process specifications	60
6.3.6	Management specifications	61
6.3.7	Specification profiles	62
6.3.8	Semantic Web and Web Services	63
6.4	SOA quality aspects	63
6.4.1	Quality requirements	64
6.4.2	The foundations of SOA quality	67
6.4.3	The key elements of SOA quality	68
6.5	References	68
7	Model-driven Architecture	71
7.1	The MDA standards	72
7.1.1	Unified Modelling Language	72
7.1.2	Meta-Object Facility	72
7.1.3	XML Metadata Interchange	73
7.1.4	Common Warehouse Meta-model	73
7.2	The basic concepts	73
7.2.1	Model	73
7.2.2	Meta modelling architecture	74
7.2.3	Architecture	75
7.2.4	Platform	75
7.2.5	Platform independence	75
7.2.6	Platform model	75
7.2.7	Model transformation	76
7.2.8	Implementation	76
7.2.9	Views and viewpoints	76
7.2.10	MDA models	76
7.3	The MDA Process	77
7.3.1	Analysis and modelling (PIM)	77
7.3.2	Modeling and design (PSM)	78
7.3.3	Implementation and deployment	79
7.4	Transformations	80
7.5	References	81
8	Grid technologies	83
8.1	Introduction	83
8.2	Grid Architecture	83
8.2.1	GLOBUS	85
8.2.2	Seamless Thinking Aid	85
8.2.3	LSF	85
8.2.4	UNICORE	86

8.2.5 Legion	86
8.2.6 Jini	86
8.3 OGSi	86
8.3.1 WSRF (Web Services Resource Framework)	87
8.4 References	88
9 Wireless networks and devices	89
9.1 Wireless Networks	89
9.1.1 Bluetooth	89
9.1.2 ZigBee	89
9.1.3 Wi-Fi	91
9.1.4 HomeRF	93
9.1.5 Z-Wave	96
9.1.6 Wireless USB	97
9.1.7 Ultra-Wideband (UWB)	97
9.2 Services Discovery	99
9.2.1 Bluetooth	99
9.2.2 Jini	100
9.2.3 SLP	101
9.2.4 Universal PnP	101
9.2.5 HAVi	103
9.2.6 Salutation	104
9.2.7 Comparison between different service discovery systems	105
9.3 Wireless Devices	106
9.3.1 RFID tags	107
9.3.2 Active Badges	108
9.3.3 GPS-enabled devices	109
9.4 References	111
10 Privacy and security	113
10.1 Privacy at the middleware and application layers	113
10.1.1 Privacy solutions at the middleware/application layers	113
10.1.2 Solutions for middleware/application native privacy threats	116
10.2 Privacy in the network	120
10.2.1 Anonymous networking	120
10.2.2 Wireless networks enhancement	122
10.2.3 RFID systems	124
10.3 Keys management, authentication and accounting	127
10.3.1 Key management	127
10.3.2 Security services and AAA	129
10.4 References	130
11 Conclusions	137
11.1 Embedded, autonomic AmI Architecture	137
11.2 Wireless Networks and devices	137
11.3 SoA and MDA middleware	138
11.4 Trust, privacy and security	139

1 Introduction

Since available technologies represent technological opportunities for the project to develop a solution meeting the project objectives, knowledge of the state-of-the-art of the technologies play an important role, including outlining technological boundaries within which the project outcomes can be positioned. Therefore, not considering the knowledge of current leading-edge technologies can seriously limit the project output in terms of functional or non-functional properties.

Based on this assumption, the purpose of this report is to provide the project consortium with information on a broad range of technologies which can be (potentially) applied in the project. In this way, the report can support design of technical solution, and subsequently improve usability as well as market potential of the project outcome.

The presented report was written as a deliverable within the workpackage WP2 "Iterative user requirements engineering". Since one of the objectives of this workpackage is "to maintain a continuous study of the technological developments affecting the Hydra middleware", the report represents an initial version of the technology watch. It is expected to provide a technology map valid at the time of writing the report (October 2006). The report provides a state of the art description of the technologies identified as relevant to the HYDRA implementation, i.e. technologies that can contribute to (or can have impact on) the implementation of the required HYDRA functionality.

The scope of the report is based on the description of the state-of-the-art provided in the Technical annex of the Project Contract. Technology areas presented in the Technical annex are covered in this report as well but this is a self-standing report (i.e. familiarity with the Technical annex is not necessary for a reader of this report). In comparison with the Technical annex, descriptions of technologies are considerably enriched, information on the technologies is updated, and the list of technologies included in the report was extended.

On the other hand, the content of the report represents current level of knowledge possessed by the project partners. In this way, it reflects knowledge distributed within the project consortium and can serve as a means, which can be used also for identification of knowledge gaps within the consortium.

2 Executive summary

The presented deliverable D2.2 "Initial Technology Watch Report" provides an overview of existing technologies, which were selected as relevant to the Hydra project. The scope of the project therefore plays the role of a context within which the technologies could be evaluated to assess whether they have a potential to contribute to the project goals. Issues, related to the purpose and scope of the report, are discussed in Introduction.

Subsequently, several sections are devoted to the selected technologies, each section focusing on one type of technologies, providing information on available technologies and discussing various issues relevant to the technologies.

Since the report's position regarding Ambient Intelligence is a vision of human beings surrounded by electronic artefacts and environments, ambient intelligence technologies are expected to combine concepts of ubiquitous computing and intelligent systems putting humans in the centre of technological developments. Keeping this vision in mind, Section 3 presents middleware platforms which are context-sensitive and providing suitable abstractions for dealing with heterogeneity and distribution while treating available resources. To complement the platforms, the section focuses on issues related to interaction, e.g. interactive devices, interaction paradigm and distribution.

In several aspects the Hydra project builds on the recently emerged idea of the Semantic web – a "web for machines", promising the opportunity for finding and processing information based on employing semantic technologies enabling expression of the semantics of the information. In addition to an overview of languages, which are able to represent a meaning, Section 4 provides a list of different formalisms for modelling web services. In order to support the use of these formalisms, the section covers frameworks and tools for semantic web services as well.

Utilising knowledge requires the presence of a scheme enabling to organise and manipulate with the knowledge. Ontologies have proven their position in this field as widely used knowledge models providing the opportunity to operationalise knowledge embedded in these models. To explore the underlying technologies, Section 5 focuses on ontology languages and issues related to ontology reasoning. Since the Hydra project is expected to target dynamic environments, the section devotes its attention to questions of ontology evolution and versioning. In order to manage changes in knowledge, the section outlines technologies for communicating the changes and version control.

Although the concept of Service Oriented Architectures is not new and has been in use already for several years, the features offered by the concept (e.g. loose coupling, abstraction from the internal design of services, dynamic discovery, platform independence, etc.) represent characteristics the Hydra project can profit from. Focus of Section 6 is on the key elements of the architecture and the implementation of the general principles in the form of Web services. The section outlines various technologies representing building blocks of Web services, e.g. service description, service discovery, service security, business process specification, and web service management.

The promise of Model-driven Architecture is to facilitate the creation of machine-readable models with a goal of long-term flexibility. Since writing platform specific code is replaced by generating the code by transformations, it enables to design models that are independent of the target platform. Section 7 introduces basic concepts and lists technologies the architecture is built on. The section discusses various issues connected with modelling and meta-modelling, platform for middleware developers, model transformations, and the modelling process.

The term Grid refers to technologies and infrastructure that enable coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. To introduce technologies used to develop grid-based systems and applications, Section 8 focuses on grid architecture and presents several environments and tools available to implement the concept of the grid architecture, including those employing the technology of web services.

Since positioning and location detection represents an important part of sensed contextual information, wireless devices are important for context sensing as an important issue constraining the definition of the Hydra middleware. The field of wireless networks is covered by Section 9. In

addition to the networks, the section discusses issues related to service discovery and introduces a few types of wireless devices.

One of the most important issues to be addressed while designing any real application is privacy and security. Section 10 focuses on privacy at the middleware/application layers (it presents several privacy solutions for access control, policy languages, and identity management) and solutions for middleware/application privacy threats. In addition, the section provides information on technologies enabling privacy in networks with a special emphasis on wireless networks. Attention is also paid to methods for key management, authentication and accounting.

The last section is dedicated to prognosticating advancements expected from the research and development activities undertaken in the HYDRA project in the technology areas described in sections providing outline of different technologies.

3 Embedded ambient intelligence

3.1 Introduction

Ambient Intelligence (AmI) is a new research field aiming at building digital environments that are aware of the humans' presence, their behaviours and needs. Among the key features, context awareness plays the most important role in any ambient intelligence system. Ambient intelligence involves the convergence of several computing areas. The first is ubiquitous or pervasive computing. Its major contribution is the development of various ad hoc networking capabilities that exploit highly portable or else numerous, very-low-cost computing devices. The second key area is intelligent systems research, which provides learning algorithms and pattern matchers, speech recognition and language translators, and gesture classification and situation assessment. A third element is context awareness; research on this problem lets us react to the situational context of people, places, and things including position and type of activity. Finally, an appreciation of the social interactions of objects in environments is essential.

AmI is the vision that technology will become invisible, embedded in our natural surroundings, present whenever we need it, enabled by simple and effortless interactions, attuned to all our senses, adaptive to users and context and autonomously acting. High quality information and content must be available to any user, anywhere, at any time, and on any device.

The AmI paradigm can be realised only through a number of technologies, all involving modern computing hardware and software. In particular, an AmI system requires the use of distributed sensors and actuators to create a pervasive technological layer, able to interact transparently with a user, either passively by observing and trying to interpret what the user actions and intentions are, but also actively, by learning the preferences of the user and adapting the system parameters (applied to sensors and actuators, for instance) to improve the quality of life and work of the occupant.

These visions show that the AmI paradigm aims to take the integration provided by the Ubiquitous Computing paradigm one step further by realizing unobtrusive environments in which many networked devices will be moved into the background and services provided will be autonomous, sensitive and responsive to the presence of people [2]. Aarts and Marzano summarize the five key technology features that characterize an AmI system [1]:

- *Embedded.* Networked devices are integrated into the environment.
- *Context aware.* System recognizes people and their situational context.
- *Personalized.* System can tailor itself to meet people's needs.
- *Adaptive.* System can change in response to people.
- *Anticipatory.* System anticipates people's desires without conscious mediation.

From that point of view, Ambient Intelligence represents a vision of the future where people are surrounded by electronic artefacts and environments, sensitive and responsive. Ambient intelligence technologies are expected to combine concepts of ubiquitous computing and intelligent systems putting humans in the centre of technological developments. Key concepts of AmI are:

- Ubiquitous Computing: that is wired, wireless and ad-hoc networking that exploit highly portable or else numerous, very-low-cost computing devices, discovery mechanisms, software architectures, system integration and prototyping, portable devices
- Context Awareness: sensors, tracking and positioning, smart devices, wearable, models of context of use, software architectures for multi platform interfaces;
- Intelligence: learning algorithms, user profiling, personalisation and adaptivity, autonomous intelligence, agent based user interfaces,

- Natural user-system interaction: ambient interfaces, multimodal interaction, innovative interaction styles and concepts.
- Appreciation of the social interactions of objects in environments, and the cultural values the new environments contribute to determine.

Ambient intelligence aims at providing ubiquitous, transparent and intelligent electronic services. These services are diverse and distributed in the user's environment. A key feature of ambient intelligence is transparency on service provisioning. The process of discovering and invoking relevant services should be hidden from the users' point of view. In order to realize this scenario, mechanisms are needed to provide smart service discovery based on the current situation of the user (e.g., user's location, his interest, user's environment characteristics, etc). Contextual information of the user is therefore an essential aspect to accomplish transparency in the service discovery process within the ambient intelligence scenario. Most of the existing service discovery mechanisms retrieve services descriptions that contain particular keywords from the user's query. In the majority of the cases this leads to low recall and low precision of the retrieved results. The reason for the first is that query keywords might be semantically similar but syntactically different from the terms in service descriptions, e.g. 'buy' and 'purchase' (synonyms). The reason for the second is that the query keywords might be syntactically equivalent but semantically different from the terms in the service description, e.g. 'order' in the sense of proper arrangement and 'order' in the sense of a commercial document used to request supply of something (homonyms). Another problem with keyword-based service discovery approaches is that they cannot completely capture the semantics of user's query because they do not consider the relations between the keywords. One possible solution for this problem is to use ontology-based retrieval. In this approach, ontologies are used for classification of the services based on their properties. This enables retrieval based on service types rather than keywords. Another drawback of the existing service discovery approaches is that the query service matching score is calculated taking into account only the keywords from the user's query and the terms in the service descriptions. Thus, regardless of the context of the user and the context of the service providers, the same list of results is returned in response to a query. By definition, context is a situation of an entity (person, place or object) that is relevant to the interaction between a user and an application. Therefore, considering the context in the query-service matching process can improve the quality of the retrieved results. However, contextual information is highly interrelated and has many alternative representations what makes it difficult to interpret and use. One possible solution is again to use ontologies to specify the interrelations among context entities and ensure common, unambiguous representation of these entities.

Context and context-awareness are central issues to AmI. The availability and use of context in interactive applications offer new possibilities to tailor applications and systems "on-the-fly" to the current situation [3].

3.2 Context-aware middleware

The role of middleware is to provide an additional layer of abstraction suitable for a specific type of applications. The intended type of applications might vary from any type of "distributed system", or as narrow as "agents in Java". However, middleware is normally intended for a specific type of distributed system, and even though a given middleware systems claims to be suitable for general distributed systems, underlying assumptions often implies certain limitations for the usefulness of the system.

In traditional distributed systems, the goal of the middleware has been to hide heterogeneity and distribution by providing ways of treating remote resources as if they were local. In wired, static environment, this has proven useful, but in dynamic, wireless environments it breaks down, since applications often need to base decisions on information about distribution and the environment. Instead, middleware systems for Ambient Intelligence focus on providing suitable abstractions for dealing with heterogeneity and distribution without hiding them, and in some cases even provide information about distribution and heterogeneity as context information.

In the following we outline central context-aware middleware platforms.

3.2.1 Aura

Aura [21][22][23][24] is a task oriented system for infrastructural environments. It runs on top of an ordinary desktop operating system, and explores the notion of *personal aura*, a system which supports the user in performing tasks. Services for management of tasks, applications, and context are provided. Unlike most other context-aware middleware, Aura does not support building applications, but instead relies on existing applications to act as *service providers* for services like text editing or playing sounds.

Tasks are controlled by a *Task Manager*, which handles migration of tasks, while services are provided by an *Environment Manager*. Tasks are abstract representations of a collection of services comprising the task. When a user moves from one environment to another, the representation of the task is moved, and service providers for the task are instantiated at the new location.

Aura provides a *Context Observer* to manage context. Context information is used to derive the *intent* of the user. The context observer merely collects context, and reports changes to the Task and Environment Managers. Depending on the detail of the collected context, the Context Observer might be able to derive the current task, location, and intent of the user. The current task is used for proactively loading the current task, while location is used to migrate tasks e.g. from the home to the office of the user. The intent is used for both tasks and location. If the user is working at home, but has a meeting scheduled at 10am and leaves the home computer, the Context Observer might derive that the user is leaving for the office, and migrate the current task without intervention from the user. If the Context Observer is unable to derive the location or intention of the user, e.g. because of insufficient sensors in the environment, the user must explicitly indicate this to Aura.

3.2.2 CARMEN

CARMEN [25] is intended for handling resources in wireless settings assuming temporary disconnects. It uses *proxies*, mobile agents residing in the same CARMEN environment as the user. If a user moves to another environment the proxy will migrate using wired connections. Each mobile user has a single proxy, which provides access to resources needed by the user. When migrating, the proxy makes sure that resources are also available in the new environment. This can happen by: moving the resources with the agent, copying the resources, using remote references, or re-binding to new resources which provide similar services. The method is determined by inspecting the *profile* of the device.

Each entity in CARMEN is described by a *profile*. *User profiles* contain information about preferences, security settings, subscribed services etc. *Device profiles* define the hardware and software of devices. *Service component profiles* define the interface of services and *Site profiles* group together the profiles which all belong to a single location. Thus, context information in CARMEN describes the entities which make up the system.

3.2.3 CARISMA

CARISMA [26][27][28] deals with adoption of middleware depending on the needs of the applications.

Profiles for each application are kept as meta-data of the middleware and consists of *passive* and *active* parts. The passive parts define actions the middleware should take when specific context events occurs, such as shutting down if battery is low. The active information defines relations between services used by the application and the policies that should be applied to deliver those services. The active part is thus only used when the application requests a service.

Different environmental conditions may be specified, which determine how a service should be delivered. At any time, the application can use reflection to alter the profile kept by the middleware through an XML representation.

To deal with conflicts between profiles, CARISMA adopts a micro-economic approach [28], where a computing system is modelled as an economy where consumers makes a collective choice over a limited set of goods. In this case, the goods are the *policies* used to provide services, not the

resources providing them. The middleware plays auctioneer in an auctioning protocol, where each application submits a single, sealed bid on each alternative profile. The auctioneer then selects the alternative which maximises the sum of bids. To determine the bid each of the applications is willing to pay, functions which translate from profile requirements to values are defined. Like profiles, these functions may be changed at any time through reflection. This type of protocol makes sense because CARISMA delivers the *same* service to all participants.

3.2.4 CoBrA (Context Broker Architecture)

CoBrA [8] is an agent-based architecture for supporting context aware computing in intelligent spaces. CoBrA goes one step further than Context Toolkit as it uses the Semantic Web languages to define ontologies of context which provide an explicit semantic representation of context that is suitable for reasoning and knowledge sharing. Central to CoBrA architecture is the presence of an intelligent context broker that maintains and manages a shared model of contexts on the behalf of a community of agents. An intelligent meeting room system called EasyMeeting has been implemented using this architecture. An ontology called COBRA-ONT was created for modelling context in an intelligent meeting room. Basic inference based on context information about meeting attendees and the devices can be made with this system.

3.2.5 Context Toolkit

Context Toolkit [6] is an architecture developed to support the building of context-aware applications. The Context Toolkit contains a combination of features (capture and access of context, storage, distribution, and independent execution from applications) and three types of abstractions (widgets, aggregators and interpreters) [6]. The context widget is responsible for acquiring context information and makes it available to applications regardless of how it is actually sensed. Context widgets automatically store all the context they sense and make this history available to any interested applications. Applications can use historical context information to predict the future actions or intentions of users. This prediction or interpretations functionality is encapsulated in the context interpreter abstraction. Context aggregators aggregate or collect context [7].

3.2.6 Cooltown

The Cooltown project [29][30] is intended to support wireless, mobile devices to interact with a web-enabled environment. The basic principle is that devices, people, and things have a web-presence identified by a URL, which provides a "rich" interface to the entity. Users interact with the web-enabled environment using PDAs to interact with the available web-services. As such, Cooltown expects wireless Internet access when users interact with the system. URLs are passed between devices in local device to device interaction. E.g. a projector might receive a presentation by receiving a URL to the file.

Context in the system is closely tied to the physical environment. For example, an infrared beacon at the entrance of a room will emit a URL which points to the page of the room [29]. When a PDA loads this page, the PDA acts as an interface to the room, thus changing behaviour based on location context. Other types of context might be used by web-applications by providing web-applications with other context like time or activity. The main principle in the collection of context is that it is provided by web-clients. Depending on which sensors the clients have, web interfaces can adapt to the context they provide. Types of context about the physical world includes [30]: *where*, *when*, *who*, *what*, and *how*.

The context is integrated with a model of the physical world, consisting of places, people and things, and relationships between them. Relationships include: *Contains*, *isContainedIn*, *isNextTo*, and *isCarriedBy*, and the list is extensible. Relationships are directional, so like hyperlinks they can be navigated in one direction, making them suitable for presenting as web pages. Relationships have properties, and can be subtypes of other relationships. The state of the model is updated automatically by sensing mechanisms ranging from infrared beacons to GPS.

The main modules in the architecture are: *Web Presence Manager, Description, Directory, Discovery modules, Autobiographer, Observer, and Control.*

Besides these modules, Cooltown offers tools to build web-presence services (applications).

3.2.7 CORTEX

The CORTEX project [31][32] is concerned with research other than context-aware middleware, but has proposed a middleware to deal with "Autonomous mobile physical objects that cooperate with other objects, either mobile or static, by capturing information in real-time from sensors event messages propagated in a MANET" [32].

The middleware is based on sentient objects. A sentient object senses and views the behaviour of neighbouring objects, reasons about them, and manipulates physical objects accordingly. Sentient objects dynamically discover each other, and share context information.

To support sentient objects, CORTEX provides a middleware based on component frameworks, each of which provides a service to the sentient objects: *Publish-Subscribe, Group Communication, Context, and QoS management.*

Publish-Subscribe is used for discovery, while the other component frameworks support communication, context retrieval and inference, and arbitration of resource allocation.

The resulting middleware is configured at deployment time and can be reconfigured at run-time through a reflective API to adapt to changes in the environment.

3.2.8 Gaia

The Gaia Operating Systems [33][34] is intended to be a *meta-operating system*. That is, a distributed middleware system providing functionality similar to an operating system. Gaia builds on the notion of an *active space*, coordinating heterogeneous devices in a physical space, typically a single room. Like operating systems, it provides: *program execution, I/O operations, file-system access, communications, error detection, and resource allocation.*

Program execution is supported by the *component manager core*, which allows any application to upload components to any node of execution in the active space. I/O operations are supported by device drivers on each node, and the functionality is exported to the rest of the active space using distributed objects. Gaia utilises the *Context File-System*, which stores files based on representation of the context. Both synchronous and asynchronous communication is supported through RPC and events. Applications can register for event notification in case of errors, and react accordingly. Finally, Gaia manages resources throughout the active space.

Gaia is structured like traditional file systems with a kernel providing the necessary services and applications built with an application framework on top.

Gaia differentiates between location, context, and events and although they can all be seen as different kinds of context, they are handled by different components. Context is collected by *context providers* and higher level context, such as activity, can be inferred from low level context. An additional presence service deals with which entities are present in an active space. Four basic types of entities are supported: application, service, device, and person.

Context is represented by first-class predicates and more complex context is represented by first-order logic operations, such as *and* and *or*. Applications are notified of changes in context through events, and can react accordingly.

3.2.9 Hydrogen Context-Framework

In the Hydrogen project [9] at Johannes Kepler University Linz, issues related to building context-aware systems using mobile devices in particular (e.g. limitations of network connections, limited computing power, and characteristics of mobile users) are addressed. The Adapter Layer is responsible to get information from sensors about the physical context, possibly enriches this

information with logical context information and delivers it to the Management Layer. This layer permits a sensor's concurrent use by different applications. The ContextServer embedded in the Management Layer provides simple methods for the application for retrieving or subscribing to a context. Applications, which use the context provided by the underneath layers, are part of the Application Layer.

3.2.10 Middlewhere

MiddleWhere [35] provides advanced location information to applications and incorporates a wide range of location sensing techniques in a model for location. Location information originates in *Location Providers* and is stored in a spatial database. A reasoning engine uses the location information from different providers to determine location and a location service uses the spatial database and the reasoning engine to provide location with a certain probability.

The location model is hierarchical and deals with three different kinds of location: *points*, *lines*, and *polygons*. Each is represented by coordinates and a symbolic name. Location is represented as GLOBs (Gaia Location Byte-string). For example, a desk could be represented as Building1/3/338/Desk1 or as Building1/3/338/(2,4,0), meaning that Desk1 in room 338, floor 3 of the Hopper building is located at coordinates (2,4,0) with respect to the coordinate system of the room. The room will have coordinates with respect to the floor, the floor with respect to the building, and the building with respect to global coordinates. In this case, the desk is represented by a point. Polygons are used for representing rooms, hallways or spaces within rooms, while lines can be used for representing doors between two rooms.

The system deals with quality of the location information. The quality is measured according to *resolution*, *confidence*, and *freshness*. Resolution differs widely between different location sensing techniques. For example, a person using a card-reader to enter a room will tell the system that the person is somewhere inside the room while GPS has a resolution down to perhaps 10 meters. An RF badge might have a resolution of 1 meter. Confidence is a measure of how precise the sensor is in terms of probability that the object is within the sensed area. This probability originates in the sensors which register the object and in the case of multiple sensors, the information is fused to yield a single value. Freshness is based on the time since the last sensor reading, and each type of sensor has an associated temporal degradation function which, based on freshness, degrades the confidence in the information.

3.2.11 MobiPADS

MobiPADS [36] is a middleware system for mobile environments. The principle entity is *Mobilets*, which are entities that provide a service, and which can be migrated between different MobiPADS environments. Each mobilet consists of a slave and a master. The slave resides on a server, while the master resides on a mobile device. Each pair cooperates to provide a specific service. Services are composed by chaining them together in specific order, and the slave mobilets on the server are nested in the same order. This provides reconfiguration based on different requirements.

MobiPADS is concerned with internal context of the mobile devices, which is used to adapt to changes in the computational environment. Thus, context types include: *processing power*, *memory*, *storage*, *network devices*, *battery* etc. Each of these has several subtypes, e.g. *size* and *free_space* for memory and storage. Mobilets are provided with changes through context events, which they subscribe to. Higher order context is derived by an *Environment Monitor*, which subscribes to event sources and has the same characteristics as other event sources.

Adoption takes place in either the middleware based on system profiles, or by letting mobilets adapt to the events they receive. Based on the requirements in the profile, the service chains can be reconfigured to deal with e.g. a constrained environment, based on programmer provided alternatives service chains. Applications have access to reflective interfaces for context, service configuration, and adoption strategies, and can change them to obtain a different service from the middleware.

3.2.12 SOCAM

SOCAM [37] is based on the idea of using ontologies to model context. The model is then used by an interpreter to reason about context. The SOCAM architecture consists of: *Context Providers*, *Context Interpreters*, a *Context Database*, a *Service Location Service*, and *Context-aware Mobile Services*. Context Providers provide external or internal context, which can be used by the mobile services directly or by Context Provider to provide higher-order context. Externally, the Context Interpreter acts as a Context Provider. Context is represented as instances of the ontology model.

The Context Interpreter consists of a *Context Reasoner* and a *Context Database*, which contains instances of the current ontology, either from users or Context Providers. The context is updated by a triggering mechanism with different intervals. Context Providers register with the Service Location Service, thus allowing Mobile Services to locate them. The Mobile Services can obtain context either by querying the located Context Providers, or by registering for specific events. SOCAM supports *rules* for specifying which methods should be invoked on events. The rules are predefined and loaded into the Context Reasoner.

SOCAM represents context as a formal ontology as predicates in OWL. The middleware supports reasoning about context, so that high level context can be derived from observed context by the Context Interpreter. The ontologies are either a generalised ontology, or a domain specific ontology which is "bounded" with the generalised ontology, or "re-bounded" if the context changes. The domain specific ontology may, for example, be re-bounded if the context shifts from an office location to a car. It is the responsibility of the Service Locating Service to load new context ontologies when applications ask for location context.

3.2.13 Stick-e Notes

Brown [4] proposes architecture based on Post-It notes metaphor to help developers to easily create context-aware applications. A range of contexts, such as locations, time, weather, people/objects or even actions/rules to be triggered when a certain event happens, can be associated with Stick-e notes. These contexts are called trigger condition; a Stick-e note is triggered if the specified condition is met. Stick-e note architecture consists of three components [5]:

- The *triggering component* matches user's present context with the context of loaded Stick-e notes and in case there is a match triggering the Stick-e note.
- The *execution component* could be any existing program executing the note.
- The *set of sensors component* feeds periodically information to the triggering module and hiding sensor specific issues.

3.3 Emerging opportunities

Ambient intelligence is anticipated to have a profound impact on the everyday life of people in the information society and to potentially permeate a wide variety of human activities. This section discusses the potential benefits of ambient intelligence from the point of view of universal access, focusing on both interaction devices and paradigms and emerging applications. Some of the issues most relevant for people with disabilities are also pointed out.

AmI research must focus on developing user-friendly low-cost solutions with a high level of network security. This involves seamless integration of nano- and opto-electronics, natural user interfaces and integration of electronics in new computing substrates like fabrics and plastic.

3.3.1 Interactive devices

The objective of AmI is to broaden the interaction between human beings and digital information technology through the usage of ubiquitous computing devices. Conventional computing primarily involves user interfaces (UIs) such as keyboard, mouse, and visual display unit; while the large amount of ambient space that encompasses the user is not utilized as it could be. AmI on the other hand uses this space in the form of, for example, shape, movement, scent and sound recognition or

output. Sensors would adapt to a homeowner through sound, scent, shape, and movement. These information media become possible through new types of interfaces and will allow for drastically simplified and more intuitive use of devices. For the communication between the latter, wireless networks will be the dominant technology. The combination of simplified use of devices and their ability to communicate eventually results in increased efficiency for the users and, therefore, creates value, leading to a higher degree of ubiquity of computing devices.

Several areas are worth highlighting as key interface trends to watch. These include the growth of agent communication languages, the introduction of affect into the interface, and the growing focus on awareness and knowledge management, each of which we briefly describe.

- **Intelligent Interface Agents.** Advances in tools and techniques for control of knowledge rich components is being advanced by specific architectures such as the Open Agent Architecture (OAA, <http://www.ai.sri.com/~oaa>) but also by government initiatives such as the Distributed Agent Markup Language (www.daml.org) and the semantic web (www.w3.org/2001/sw, www.semanticweb.org)
- **Affective interfaces.** Recognizing and expressing mood and emotion via the interface has received increased interest. This could come, for example, in the form of detecting delight or stress via language, speech, and gesture or expressing emotional displays via an interactive life-like agents. It could also be as practical as detecting and effecting drowsiness in a car driver interface
- **Awareness.** The explosion of Instant Messaging (IM) and associated presence information has increased user desire for information regarding user identity, physical and virtual location, activity (e.g., idle, working), availability, and communication capability (e.g., platform, interactive devices, network connectivity). In addition to Awareness of individual characteristics, there also is a need for awareness of the emergence and tracking of group activity and roles participants play (e.g., who is the leader, facilitator, key contributor)
- **Knowledge Management.** Strongly related to awareness are areas necessary to support knowledge access, including:
 - **Expert Discovery:** Modelling, cataloguing and tracking of distributed organizations and communities of experts.
 - **Knowledge Discovery:** Identification and classification of knowledge from unstructured multimedia data.
 - **Knowledge Sharing:** Awareness of and access to enterprise expertise and know-how.

3.3.2 Interaction paradigm

At the same time, the way in which computing tasks are accomplished will undergo radical changes. Interaction will shift from an explicit paradigm, in which the user's attention is on computing, to an implicit paradigm, in which interfaces themselves proactively drive human attention when required. Moreover, the complexity of distributed and dynamically defined systems will not allow humans to operate devices in a step-by-step manner toward the completion of a task. Rather, humans will manage tasks by delegating their execution to intelligent computing units in the technological environment. The increased intelligence intrinsic in the environment and the more intuitive forms of interaction, if appropriately designed, will have the potential of counterbalancing, to a certain extent, the effects of motor, sensory, cognitive, and memory limitations. For example, task delegation may be expected to considerably alleviate both the physical and the cognitive efforts required for interaction. While these characteristics are of obvious benefit to all users, they can make the difference between usable and non-usable applications and services for people with disabilities.

3.3.3 Interaction distribution

Due to the intrinsic characteristics of the new technological environment, it is likely that interaction will pose different physical, perceptual, and cognitive demands on humans when compared with

currently available technology. It is therefore important when ensuring universal access to these technologies (in particular, for elderly people and people who are disabled), to investigate how human functions will be engaged in the emerging forms of interaction, and how this will affect physical interaction and an individual's perceptual and cognitive space (e.g., emotion, vigilance, information processing, memory) [12, 13]. The main challenge in this respect is to identify and avoid forms of interaction that may lead to negative consequences, such as confusion, cognitive overload, and frustration. This is particularly important given the pervasive impact of the new environment on all types of everyday activities and on the way of living. A first implication is that interactive systems must be capable of dealing in real time with the distribution of input and output in the environment in such a way as to provide humans with continuous, flexible, and coherent communication, both with the environment and with others, by proportionally using all the available senses and communication channels, while optimizing human and system resources [14]. This implies an understanding of the factors that influence the distribution and allocation of input and output resources in different situations for different individuals, taking into account possible human limitations.

3.3.4 Automation versus human control

Providing effective and efficient human control for the dynamic and distributed system will also become critical. In particular, it will be necessary to establish an appropriate balance between automated learning on the part of the intelligent environment, human behaviour patterns, and human intervention aimed at directing and modifying the behaviour of the environment. This aspect of the emerging technologies needs to be carefully taken into account, particularly when elderly people and people with cognitive disabilities are involved, as services that monitor the health status or the location of users may also interfere with their ability to make decisions [15].

3.3.5 Content and functionality

A prerequisite for the successful development of the ambient intelligence environment is that future computing needs in everyday life are appropriately anticipated [16]. An in-depth understanding of the factors that will determine the usefulness of interactive artefacts in context is required. These requirements are likely to be more subjective, complex, and interrelated than in previous generations of technology. For example, elderly people and people with disabilities will need personalized navigation services, including location-, content-, and disability-dependent accessibility information.

3.3.6 Health and safety

In a situation in which technology may act on the physical environment and deal with critical situations without the direct intervention of humans, it is likely that new health and safety hazards may emerge. Possible malfunctions or misinterpretations of monitored data can lead to unforeseeable consequences for the segments of the population that use technology to overcome human limitations and will therefore be more dependent on it than others. This implies the necessity of monitoring every relevant element in context, and identifying the elements that should be monitored by each device and the conditions and parameters according to which monitoring should take place. For example, in the health-care domain, the challenge is to optimally extract the most critical information from the patient by using a set of sensors and tasks and to present that information to a remote location in an appropriate form [17]. Furthermore, appropriate backup strategies must be elaborated. An important issue in this respect is the notion of redundancy (of information, communication channels, monitoring mechanisms, etc.), which, through cross-checking mechanisms, can contribute toward increasing the correct functioning of the technological environment and minimizing risks. A related challenge is that of interoperability among different technologies and devices, because the correct functioning of the intelligent environment as a whole needs to be ensured. Maintenance of ambient intelligence environments and of their components is also expected to play a significant role with respect to health and security issues.

3.3.7 Privacy and security

As technology becomes embedded in everyday objects and in the environment, functional and interaction aspects of technological artefacts may become subordinated to other personal factors of choice [18]. The most important ethical issue in this respect concerns privacy and anonymity and the effective protection of personal data that is collected through the continuous monitoring of people. In this respect, new challenges arise concerning how a person will be able to know when and what type of information is recorded, by whom, and for what use in a technological environment where personal information is continuously collected by numerous invisible receptors.

3.4 Conclusion

Currently, a variety of user-interface development frameworks are available which address issues of automatic adaptation. For example, model-based approaches are often oriented to the development of multiplatform user interfaces or context-based adaptation. On the other hand, development approaches targeted to accessibility for people with disabilities have also emerged [19]. The Unified User Interface Development Framework [20] is a development approach comprising architecture and a set of development tools suitable for the development of user interfaces that can automatically adapt to user characteristics and use contexts. An example of an architecture and development support tool for dynamic dialog composition in ambient computing, along with a discussion of the issues and challenges involved, is provided in [20].

3.5 References

- [1] Aarts, E. and Marzano, S. eds., "The New Everyday: Visions of Ambient Intelligence", 010 Publishing, The Netherlands, 2003.
- [2] Aarts, E., "Ambient intelligence: A multimedia perspective," IEEE Trans. Multimedia, vol. 11, no. 1, Jan.-Mar. 12-19, 2004.
- [3] Schmidt, A., "Interactive Context-Aware Systems Interacting with Ambient Intelligence", Ambient Intelligence G. Riva, F. Vatalaro, F. Davide, M. Alcañiz (Eds.) IOS Press, 2005.
- [4] Brown, P.J., "The Stick-e Document: A Framework For Creating Context-aware applications", In the Prods. of the Electronic Publishing, pp. 259-272, 1996.
- [5] Stottinger, M., "Context-Awareness in Industrial Environments", 2005.
- [6] Dey, Anind K., "Providing Architectural Support for Building Context-Aware Applications". PhD thesis, Georgia Institute of Technology, 2000.
- [7] Dey, Anind K., "Understanding and Using Context". Personal and Ubiquitous Computing Journal 5(1): pp. 4-7, 2001.
- [8] Chen, H., Finin, T., and Joshi, A., "An ontology for contextaware pervasive computing environments". Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, 2003.
- [9] Sd Chen, H., Finin, T., and Joshi, A., "An ontology for contextaware pervasive computing environments". Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, 2003.
- [10] Chen, H., Finin, T., and Joshi, A., "An ontology for contextaware pervasive computing environments". Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, 2003.
- [11] Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., and Altmann, J., "Context-awareness on mobile devices- the hydrogen approach". In Prods. of the 36th Annual HawaiiInt'l Conf. on System Sciences, pp. 292-302, 2002.
- [12] A. K. Dey, P. Ljungstrand, and A. Schmidt, "Distributed and Disappearing User Interfaces in Ubiquitous Computing," Proceedings of CHI2001 Workshop on Distributed and Disappearing UIs in Ubiquitous Computing, Seattle, WA (2001).

-
- [13] T. Prante, "Designing for Usable Disappearance-Mediating Coherence, Scope, and Orientation," Proceedings of CHI2001 Workshop on Distributed and Disappearing UIs in Ubiquitous Computing, Seattle, WA (2001).
- [14] G. Faconti and M. Massink, "Continuous Interaction with Computers: Issues and Requirements," in Universal Access in HCI: Towards an Information Society for All, Volume 3 of the Proceedings of HCI International 2001, New Orleans, LA (2001), Lawrence Erlbaum Associates, Mahwah, New Jersey, pp. 301–304.
- [15] J. Abascal and A. Civit, "Mobile Communication for Older People: New Opportunities for Autonomous Life," EC/NSF Workshop on Universal Accessibility of Ubiquitous Computing: Providing for the Elderly, Alcácer do Sal, Portugal (2001).
- [16] G. D. Abowd and E. D. Mynatt, "Charting Past, Present and Future Research in Ubiquitous Computing," ACM Transactions on Computer-Human Interaction, Special issue on HCI in the New Millennium 7, No. 1, pp. 29–58 (2000).
- [17] C. E. Lathan, "The Role of Telemedicine, or Telecare, in Rehabilitation and Home Care," EC/NSF Workshop on Universal Accessibility of Ubiquitous Computing: Providing for the Elderly, Alcácer do Sal, Portugal (2001).
- [18] T. Prante, "Designing for Usable Disappearance-Mediating Coherence, Scope, and Orientation," Proceedings of CHI2001 Workshop on Distributed and Disappearing UIs in Ubiquitous Computing, Seattle, WA (2001).
- [19] A. Savidis and C. Stephanidis, "Unified User Interface Development: Software Engineering of Universally Accessible Interactions," Universal Access in the Information Society 3, No. 3, pp. 165–193 (2004).
- [20] A. Savidis and C. Stephanidis, "Distributed Interface Bits: Dynamic Dialogue Composition from Ambient Computing Resources," in Personal and Ubiquitous Computing, Springer-Verlag, Berlin, Germany (2005).
- [21] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. "Project Aura: Toward Distraction-Free Pervasive Computing". IEEE Pervasive computing, 1(2): pp. 22–31, April–June 2002.
- [22] U. Hengartner and P. Steenkiste. "Protecting access to people location information". In SPC, pp. 25–38, 2003.
- [23] G. Judd and P. Steenkiste. "Providing contextual information to pervasive computing applications". In PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, page 133, Washington, DC, USA, 2003. IEEE Computer Society.
- [24] J. P. Sousa and D. Garlan. "Aura: An architectural framework for user mobility in ubiquitous computing environments". In WICSA 3: Proceedings of the IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture, pp. 29–43, Kluwer, B.V., Deventer, The Netherlands, The Netherlands, 2002.
- [25] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli. "Context-aware middleware for resource management in the wireless internet". IEEE Transactions on Software Engineering, 29(12): pp. 1086–1099, 2003.
- [26] L. Capra. "Mobile computing middleware for contextaware applications". In ICSE '02: Proceedings of the 24th International Conference on Software Engineering, pp. 723–724, New York, NY, USA, 2002. ACM Press.
- [27] L. Capra, W. Emmerich, and C. Mascolo. "Reflective middleware solutions for context-aware applications". Lecture Notes in Computer Science, 2192: pp. 126–133, 2001.
- [28] L. Capra, W. Emmerich, and C. Mascolo. "Carisma: context-aware reflective middleware system for mobile applications". IEEE Transactions on Software Engineering, 29(10): pp. 929 – 45, 2003/10/.
- [29] J. Barton and T. Kindberg. "The Cooltown user experience". Technical report, Hewlett Packard, February 2001.

-
- [30] P. Debaty, P. Goddi, and A. Vorbau. "Integrating the physical world with the web to enable context-enhanced services". Technical report, Hewlett-Packard, September 2003.
 - [31] H. A. Duran-Limon, G. S. Blair, A. Friday, P. Grace, G. Samartzidis, T. Sivaharan, and M. WU. "Contextaware middleware for pervasive and ad hoc environments", 2000.
 - [32] C.-F. Sørensen, M. Wu, T. Sivaharan, G. S. Blair, P. Okanda, A. Friday, and H. Duran-Limon. "A context-aware middleware for applications in mobile ad hoc environments". In MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing, pp. 107–110, New York, NY, USA, 2004. ACM Press.
 - [33] M. Román, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. "Gaia: A Middleware Infrastructure to Enable Active Spaces". *IEEE Pervasive Computing*, pp. 74–83, Oct-Dec 2002.
 - [34] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. Campbell, and K. Nahrstedt. "A middleware infrastructure for active spaces". *IEEE Pervasive Computing*, 1(4): pp. 74 – 83, 2002/10/.
 - [35] A. Ranganathan, J. Al-Muhtadi, S. Chetan, R. H. Campbell, and M. D. Mickunas. "Middlewhere: A middleware for location awareness in ubiquitous computing applications". In H.-A. Jacobsen, editor, *Middleware*, volume 3231 of *Lecture Notes in Computer Science*, pp. 397–416. Springer, 2004.
 - [36] A. Chan and S.-N. Chuang. "Mobipads: a reflective middleware for context-aware mobile computing". *IEEE Transactions on Software Engineering*, 29(12): pp. 1072 – 85, 2003/12/.
 - [37] T. Gu, H. K. Pung, and D. Q. Zhang. "A middleware for building context-aware mobile services". In *Proceedings of IEEE Vehicular Technology Conference*, May 2004.

4 Semantic web

WWW (World Wide Web) represents a huge repository of information which can be retrieved and utilised (if user is lucky enough to find what he/she needs – but it is another story beyond the scope of this report). Its success turns it into a phenomenon which in eyes of many people plays the role of the synonym of the Internet. Unfortunately, information is represented with no meaning associated – the meaning of retrieved information can be (re-)established only in the process of interpreting the information by humans. As a result, information scattered throughout the current (traditional) version of the web is almost totally useless for software, non-human users (machine agents).

In attempt to respond to this situation, the term “Semantic Web” was coined by Tim Berners-Lee and his colleagues [1] referring to a “web for machines” as opposed to a web to be read by humans. In their understanding “*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*”

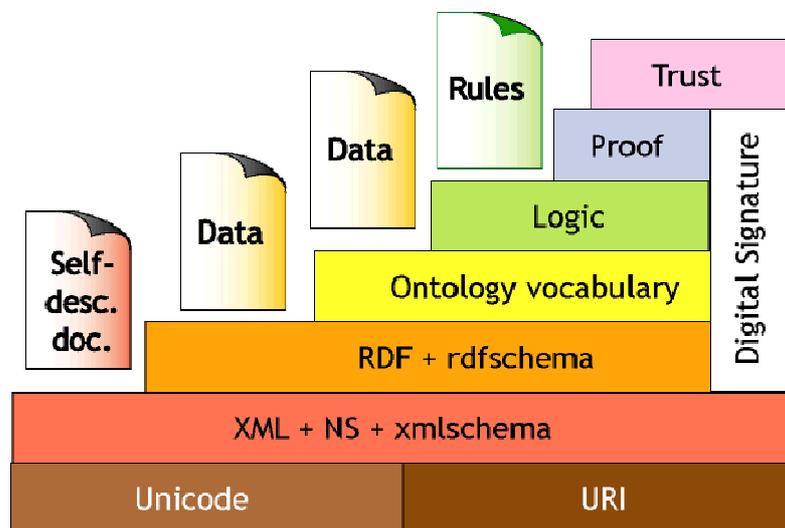


Figure 1 Semantic web layers (from [12])

The Semantic Web is the opportunity for providing, finding and processing information via the Internet with the help of machines (and mostly also for machines) which are capable of dealing with the semantics of the information. The idea is to transform information into something meaningful to actors who seek to enhance their knowledge in order to satisfy a specific concern or accomplish a specific task related to their particular context. The vision of the Semantic Web is based on the employment of semantic technologies that allow the meaning of information and the meaning of associations between information to be known and processed at execution time.

To fulfil the promises and enable semantic technologies to work, there must be a knowledge model (of some part) of the world that is used to provide meaning to information to be processed within an application. The knowledge model has the form of a semantic model which differs from other kind of models [2]:

- Meaning is represented through connectivity. The meaning of terms, or concepts, in the model is established by the way they connect to each other.
- A semantic model expresses multiple viewpoints.
- Semantic models represent knowledge about the world in which systems operate and are shared across applications.
- Several interconnected models could be used to represent different aspects.

- Use of a model is often referred to as “reasoning over the model”. The reasoning can range from a very simple process of graph search to intricate inferencing.

Although the role of a semantic model can be played by a simple taxonomy, nowadays use of semantically richer ontologies (ontological models) dominates.

Although most common definition states that “*An ontology is a specification of conceptualisation*”, more detailed definitions can make things a bit clearer. One of them states that “*The subject of an ontology is the study of the categories and things which exist in some domain. The product of such study, called an ontology, is a catalogue of the types of things that are assumed to exist in a domain of interest from the perspective of a person who talks about the domain using some language*”. From the practical point of view, an ontology is a network of connections defining explicit relationships (named and differentiated) between concepts. New knowledge can be derived by examining the connections between concepts. Simple ontologies are just networks of connections, richer ontologies include rules and constraints governing these connections.

The semantic web is not so much a technology as an infrastructure, enabling the creation of meaning through standards, mark-up languages, and related processing tools. To represent ontologies in a formal way, several languages can be used. The Semantic Web principles are implemented in the layers of Web technologies and standards (see Figure 1 [12]). The most common ontology languages are briefly described below (all the presented languages are supervised by the World Wide Web Consortium [3]).

XML

XML was widely accepted and used as a convenient information representation and exchange format. XML itself does not carry semantics, but it serves as the base syntax for the leading ontology languages. Later additions like XML-DTD (Document Type Definition) and XML-Schema, added some syntactic rules like enumerations, cardinality constraints, and data types, but still lacked even simple semantics like inheritance.

RDF

RDF (Resource Description Framework) is a standard way for defining simple descriptions. RDF supports semantics - a clear set of rules for providing simple descriptive information. RDF enforces a strict notation for the representation of information, based on resources and relations between them. The RDF data model provides three object types: resources, properties, and statements. Resource may be an object; property is a specific aspect, characteristic attribute, or relation used to describe a resource; statement is a triple consisting of two nodes and a connecting edge. The strength of the language is in its descriptive capabilities, but it still lacks some important features required in an ontology language such as inferences for example. However, ontology languages built on top of RDF as a representation and description format.

RDF Schema

RDF Schema (RDFS) enriches the basic RDF model, by providing a vocabulary for RDF, which is assumed to have certain semantics. Predefined properties can be used to model instance of and subclass of relationships as well as domain restrictions and range restrictions of attributes. Indeed, the RDF schema provides modelling primitives that can be used to capture basic semantics in a domain neutral way. That is, RDFS specifies metadata that is applicable to the entities and their properties in all domains. The metadata then serves as a standard model by which RDF tools can operate on specific domain models, since the RDFS meta-model elements will have a fixed semantics in all domain models.

OWL

OWL is the newest W3C recommendation for ontology definition. OWL enhances RDF vocabulary for describing properties and classes: relations between classes (e.g. subclasses), cardinality, equality, richer typing of properties, characteristics of properties (e.g. symmetry) and instances. OWL is quite a sophisticated language. The most important feature is its capability for description logic (DL) reasoning (Description Logics are a family of logic-based knowledge representation formalisms designed to represent and reason about the knowledge of an application domain in a structured and well-understood way). The OWL language also provides three increasingly expressive sublanguages: OWL Lite, OWL DL, and OWL Full, each offers a different level of expressiveness at the trade-off for

simplicity, thus offering a suitable sub language parts available for use according to expressivity needs.

An important constituent of the semantic web is represented by inference engines. Their aim is to reason over ontological models to prove statements or to deduce new knowledge from already explicitly presented knowledge. They are expected to make explicit those facts that are present in the ontology only implicitly. The reasoning over ontology can have the following purposes:

- *Validation.* Validating ontology means ensuring that the ontology is a good representation of the domain of discourse that should be modelled. Reasoning is extremely important for validation. For example, checking whether an ontology is internally consistent is crucial: obviously, no inconsistent theory can be a good representation of domain.
- *Analysis.* In analysis one assumes that the ontology is a faithful representation of the domain, and tries to deduce facts about the domain by reasoning over the ontology. In a sense of trying to collect new information about the domain by exploiting the ontology. Obviously, analysis can also provide input to the validation phase.
- *Harmonisation.* Myriads of ontologies can be used within the semantic web environment. Since each ontology represents a particular point of view, using different ontologies to represent meaning of information within a domain of interest results in mismatches in understanding and dealing with this information. In order to avoid it, semantic mappings between ontologies must be done.

At the beginning, the idea of the semantic web tried just to enhance the current version of the web. It started out with a document oriented approach. The basic idea was to make web pages identifiable by computers as information resources carrying not only information (readable only by humans) but the meaning of this information as well. The meaning was added by annotating these pages with semantic mark-up. Ontologies here define a shared conceptualization of the application domain at hand and provide the basis for defining metadata, that have a precisely defined semantics, and that are therefore machine-processable. The idea of semantically annotated web pages with machine-interpretable description of their content aimed at automated processes of searching and accessing pages enabling human users to better utilise information stored on the web. In addition to human users, the semantic web enables the participation of non-human users as well. These machine agents do not need to deal with whole web pages. Instead of this, they exchange chunks of data with each other. Although they can communicate using different protocols, technology of web services has become a dominant way of communication with and using services of applications in the web environment.

Formerly, the problem of interoperability of different agents was tackled by translation technologies, most commonly by field to field mapping. The semantic web enables agents to exchange chunks of data with meaning associated to the data using semantic technologies. Advanced applications can use ontologies to relate the information to a semantic model of a given domain. In this way semantic technologies offer a new way to integrate different applications. Nowadays, the field of semantic interoperability is the most addressed problem connected with the idea of the semantic web.

4.1 Formalisms for modelling web services

4.1.1 OWL-S (Web Ontology Language for Services)

OWL-S [4] is OWL ontology for semantic description of the web services. The structure of the OWL-S consists of a service profile for service discovering, a process model which supports composition of services, and a service grounding, which associates profile and process concepts with the underlying service interfaces.

Service profile has functional and non-functional properties. Functional properties describe the inputs, outputs, preconditions and effects of the service (IOPEs). The non-functional properties describe the semi-structured information intended for human users for service discovery, e.g. service

name, description and parameters which incorporate further requirements on the service capabilities (e.g. security, quality of service, geographical scope, etc.).

Service model specifies how to interoperate with the service. The service is viewed as a process which defines the functional properties of the service (IOPEs), together with details of its constituent processes (if the service is a composite service). The service model functional properties can be shared with the service profile. OWL-S distinguishes between atomic, simple, and composite processes. OWL-S atomic processes can be invoked, have no sub-processes, and are executed in a single step from the requester's point of view. The simple processes are used as elements of abstraction, they are viewed as executed in a single step, but they are not invocable. Composite processes consist of simple processes and define their workflows using control constructs, such as sequence, split, if-then-else or iterate.

Service grounding enables the execution of the web service by binding the abstract concepts of the OWL-S profile and process model to concrete messages and protocols. Although different message specifications are supported by OWL-S, the widely accepted WSDL is preferred.

4.1.2 WSMO (Web Service Modelling Ontology)

WSMO [5] is a conceptual model for describing semantic web services. It consists of four major components: ontologies, goals, web services and mediators.

Ontologies provide the formal semantics to the information used by all other components. WSMO specifies the following constituents as part of the description of ontology: concepts, relations, functions, axioms, and instances of concepts and relations, as well as non-functional properties, imported ontologies, and used mediators. The latter allows the interconnection of different ontologies by using mediators that solve terminology mismatches.

Goal specifies objectives that a client might have when consulting a web service, i.e. functionalities that a web service should provide from the user perspective. In WSMO a goal is characterized by a set of non-functional properties, imported ontologies, used mediators, the requested capability and the requested interface.

A *web service* description in WSMO consists of five sub-components: non-functional properties, imported ontologies, used mediators, a capability and interfaces. The capability of a web service defines its functionality in terms of preconditions, post-conditions, assumptions and effects. A capability may be linked to certain goals that are solved by the web service via mediators. Preconditions, assumptions, post-conditions and effects are expressed through a set of axioms and a set of shared all-quantified variables. The interface of a web service provides further information on how the functionality of the web service is achieved. It describes the behaviour of the service from the client's point of view (service choreography) and how the overall functionality of the service is achieved in terms of cooperation with other services (service orchestration). A choreography description consists of the states represented by ontology, and the if-then rules that specify (guarded) transitions between states. The ontology that represents the states provides the vocabulary of the transition rules and contains the set of instances that change their values from one state to the other. Like for the choreography, an orchestration description consists of the states and guarded transitions. In extension to the choreography, in an orchestration transition rules, that have as a post-condition the invocation of a mediator that links the orchestration with the choreography of a required web service, can also appear.

Mediators describe elements that aim to overcome structural, semantic or conceptual mismatches that appear between the different components that build up a WSMO description.

WSMO is formalized using the Web Service Modelling Language (WSML) which is based on description logic, first-order logic and logic programming formalisms.

4.1.3 WSDL-S (Web Service Semantics)

WSDL-S [6] is a small set of proposed extensions to Web Service Description Language (WSDL) by which semantic annotations may be associated with WSDL elements.

WSDL-S defines URI reference mechanisms to the interface, operation and message WSDL constructs to point to the semantic annotations defined in the externalized domain models. WSDL-S defines the following extensibility elements and attributes:

- *modelReference* element - allows for one-to-one associations of WSDL input and output type schema elements to the concepts in a semantic model;
- *schemaMapping* attribute - allows for many-to-many associations of WSDL input and output complex type schema elements to the concepts in a semantic model. It can point to a transformation (for example XSLT) from XML data to the external ontological data in RDF/OWL or in WSML;
- *precondition* and *effect* elements - are used on WSDL interface operations to specify conditions that must hold before and after the operation is invoked. The conditions can be specified directly as an expression with format defined by the semantic language or by reference to the semantic model;
- *category* element - provides a pointer to some taxonomy category. It can be used on a WSDL interface and is intended to be used for taxonomy-based discovery.

4.1.4 BPEL4WS (Business Process Execution Language for Web Services)

BPEL4WS [7] is a specification that models the behaviour of web services in a business process interaction. It is based on the XML grammar which describes the control logic required to coordinate web services participating in a process flow. An orchestration engine can interpret this grammar, thus it can coordinate activities in the process. BPEL4WS is a layer on the top of WSDL (Web Services Description Language). WSDL defines the specific operations and BPEL4WS defines how the operations can be sequenced. Every BPEL4WS process can be considered as a web service using WSDL describing the public entry and exit points for the process. WSDL data types are used within a BPEL4WS process to describe the information that passes between requests. WSDL might be used to reference external services required by the BPEL4WS process. BPEL4WS provides support for both executable and abstract business processes. The executable process models a private workflow. The abstract process specifies the public message exchanges between parties. The executable processes provide orchestration support while the business protocols (abstract processes) focus more on the choreography of the services.

Support for basic and structured activities is included. The basic activities might be receiving or replying to message requests as well as invoking external services. The structured activities specify what activities should run in what order – the whole process flow. These activities also provide support for conditional looping and dynamic branching. The structured activities might specify that certain activities should run sequentially or in parallel. *Containers* and *partners* are two important elements within BPEL4WS. A container is a variable for exchange in the message flow. A partner could be any service that the process invokes or any service that invokes the process. Each partner is mapped to a specific role that it fills within the business process. This is managed by containers.

In BPEL4WS, a set of activities can be grouped into a single transaction – it means that the steps enclosed in the scope should either all complete or all fail. Within this scope, the developer can then specify compensation handlers that should be invoked if an error occurs. BPEL4WS provides a robust exception handling mechanism through the use of throw and catch clauses, similar to the Java programming language.

4.2 Frameworks and tools for semantic web services

4.2.1 OWL-S Tools

A set of disparate OWL-S tools [8] exists, but not a complete execution environment based on OWL-S concepts. Instead of it, the tools have to be integrated by user. The set includes editor, matchmaker and annotator (several additional tools exist).

OWL-S editor is divided into three main parts: creator, validator and visualiser. The creator enables to create an empty OWL-S description either from a template or through a wizard. The validator part serves for validating the URIs used in the OWL-S descriptions and also validates the syntax of the ontologies. The visualiser part enables the user to visualise the descriptions and service compositions in a graphical manner by exploiting UML activity diagrams.

DAML-S Matchmaker is a Web Service that helps make connections between service requesters and service providers. The Matchmaker allows users or software agents to find each other by providing a mechanism for registering service capabilities. It calculates the syntactical and semantic similarity among service capability descriptions. The matching engine of the matchmaking system contains five different filters for namespace comparison, word frequency comparison, ontology similarity matching, ontology subsumption matching, and constraint matching.

ASSAM (Automated Semantic Service Annotation with Machine learning) WSDL Annotator is an application that assists the user in annotating Web Services. Annotations can be exported in OWL-S. WSDL files can be annotated with an OWL ontology with a point-and-click-interface, but the key feature is machine learning assisted annotation.

4.2.2 WSMX (Web Service Execution Environment)

WSMX [9] is an execution environment which enables discovery, selection, mediation, and invocation of Semantic Web Services. WSMX is based on the conceptual model provided by WSMO, being at the same time a reference implementation of it. It is the scope of WSMX to provide a test bed for WSMO and to prove its viability as a mean to achieve dynamic interoperability of Semantic Web Services.

Nowadays, some modules are not implemented or have limited functionality. The main components that have been already designed and implemented in WSMX are: core component, resource manager, discovery, data and process mediator, communication manager, choreography engine, and web service modelling toolkit.

Core component is the central component of the system connecting all the other components and managing the business logic of the system. Resource manager manages the set of repositories responsible for the persistence of the WSMO and non-WSMO related data flowing through the system. Discovery component has the role of locating the services that fulfil a specific user request. This task is based on the WSMO conceptual framework for discovery which envisions three main steps in this process: goal discovery, web service discovery, and service discovery. Currently, the service discovery covers only the matching of user's goal against service descriptions based on syntactical consideration.

Two types of mediators are provided by WSMX to resolve the heterogeneity problems on data and process level. Data mediation is based on paradigms of ontology mappings and alignment with direct application on instance transformation. The process mediation offers functionality for runtime analysis of two given patterns (i.e. WSMO choreographies) and compensates the possible mismatches that may appear.

Communication manager through its two subcomponents, the receiver and the invoker, enables the communication between the requester and the provider of the services. Choreography engine has to provide a means to store and retrieve choreography interface definitions, to initiate the communication between the requester and the provider in direct correlation with the results returned by the process mediator, and to keep track of the communication state on both the provider and the requester sides.

The web services modelling toolkit is a framework for rapid creation and deployment of homogeneous tools for semantic web services. An initial set of tools includes a WSML editor for editing WSML and publishing it to WSMO repositories, a monitor for monitoring the state of the WSMX environment, a data mediation tool for creating mappings between ontologies, and a management tool for managing the WSMX environment.

Even if the reasoner is not a part of the WSMX development effort, a WSML compliant reasoner is required by various components such as data mediator, process mediator and discovery.

4.2.3 IRS (Internet Reasoning Service) III

IRS [10] is a framework for Semantic Web Services that supports the publication, location composition and execution of Web Services based on their semantic descriptions. IRS supports the conceptual model defined by WSMO and also provides mappings for service descriptions provided in OWL-S. Although the approach is quite competitive to WSMX, choreography and orchestration do not follow WSMO specification and they are implemented in a non-standard way.

The main components of IRS are the IRS server, the IRS publisher and the IRS client. The server stores the descriptions of goals, mediators and web services along with domain ontologies. Discovery, composition, mediation, reasoning and invocation are all controlled by the server. Finally, the client provides a user-interface for goal-based web service invocation.

The publisher carries out the tasks required for publication. Publication has two roles in IRS. The first is where a web service represented by a URI endpoint is associated with a semantic service description known to IRS. The second is where standalone Java or Lisp code is wrapped to make it appear as a web service and then, as in the first case, the service is associated with a semantic service description known to IRS. Once a service has been published to IRS it is available to be used in the achievement of a user goal.

IRS has its foundation in an earlier IBROW project which made the distinction between tasks that need to be solved and problem solving methods that "provide abstract, implementation-independent descriptions of reasoning processes which can be applied to solve tasks in specific domains". Adopting the WSMO conceptual model, tasks in IRS are modelled as goals while problem solving methods are modelled as services. Discovery in IRS is based on matching the pre-conditions and post-conditions defined in the semantic descriptions of goals and services known to the IRS server.

4.2.4 METEOR-S

METEOR-S [11] project proposes the application of semantics to existing web service technologies. In particular the project endeavours to define and support the complete life cycle of semantic web service processes. The project extends WSDL to support the development of semantic web services using semantic annotation from additional type systems such as WSMO and OWL ontologies. It is not based on an overall conceptual model and it is rather a collection of related discrete tools than a single, encapsulated architecture.

The development module provides a GUI based tool for creating semantic web services using WSDL-S. The tool provides support for semi-automatic and manual annotation of existing web services or source code with domain ontologies. The publication and discovery module provides support for semantic publication and discovery of web services. It provides support for discovery in a federation of registries as well as a semantic publication and discovery layer over UDDI. The composition module consists of two main sub-modules - the constraint analysis and optimization sub-module (it deals with correctness and optimization of the process on the basis of quality service constraints) and the execution environment. The execution environment provides proxy-based dynamic binding support to BPWS4J execution engine for BPEL4WS.

The current implementation of METEOR-S allows for the creation of WSDL-S descriptions from annotated source code, the automatic publishing of WSDL-S descriptions in enhanced UDDI registries, and the generation of OWL-S descriptions, from WSDL-S, for grounding, profile and service.

4.3 References

- [1] Berners-Lee T., Hendler J., Lassila O.: "The Semantic Web. A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities". Scientific American, 284 (5), pp. 34-43, May 2001. <http://www.sciam.com/>
- [2] Semantic Technology. TopQuadrant Technology Briefing, March 2004, http://www.topquadrant.com/tq_white_papers.htm
- [3] The World Wide Web Consortium. <http://www.w3.org/>

- [4] Web Ontology Language for Services (OWL-S), <http://www.daml.org/services/owl-s/1.1/>
- [5] Web Service Modelling Ontology (WSMO), <http://www.wsmo.org/TR/d2/v1.1/>
- [6] Web Service Semantics (WSDL-S), <http://www.w3.org/Submission/WSDL-S/>
- [7] Business Process Execution Language for Web Services (BPEL4WS).
<http://www-128.ibm.com/developerworks/library/specification/ws-bpel/>
- [8] OWS-S Tool Directory. <http://www.daml.org/services/owl-s/tools.html>
- [9] Web Service execution Environment. <http://www.wsmx.org/>
- [10] Internet Reasoning Service. <http://kmi.open.ac.uk/projects/irs/>
- [11] Semantic Web Services and processes. <http://lsdis.cs.uga.edu/projects/meteor-s/>
- [12] W3 org at <http://www.w3.org/2001/12/semweb-fin/w3csw>

5 Ontology-based knowledge modelling

5.1 Ontology

5.1.1 Ontology languages

Ontology is widely accepted as conceptualization of a domain of interest that can be used in several ways to model, analyze and reason upon the domain. From semantic web point of view ontologies are metadata schemas, providing a controlled vocabulary of terms, each with an explicitly defined and machine processable semantics. Figure 2 shows the illustration example of general ontology for semantic description of device [40]. The information related to a device is logically divided into five classes depending on the type of information they provide. Device description contains basic information related to a device such as the device name, vendor details and the model of the device. Hardware description contains the details about the hardware resources of the device, the details of its CPU, the connection to the network and memory. Software description contains the details of the operating system of the device where relevant. The device status contains the details of its location, CPU usage and the power. Location details will be required when service selection needs to consider the location of the device in choosing the right service.

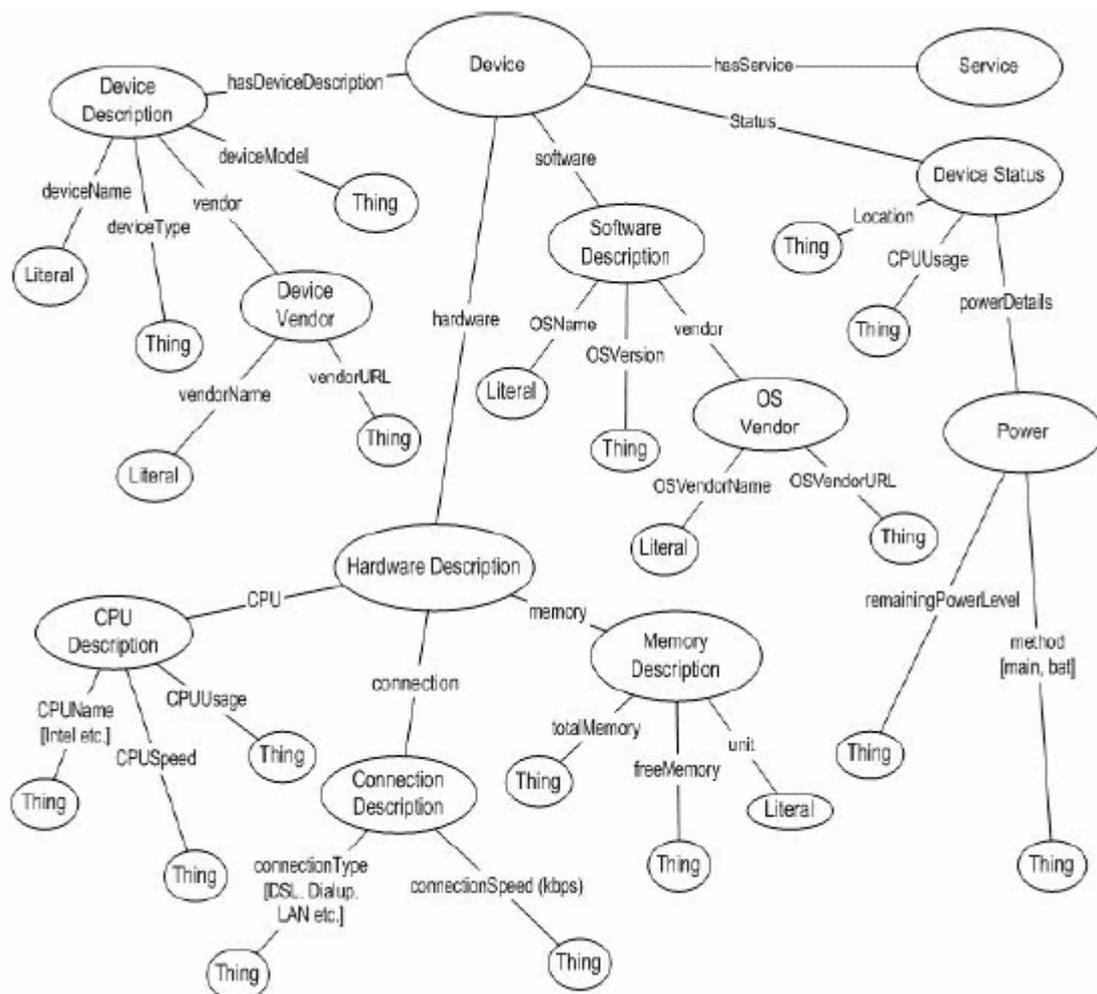


Figure 2 Illustration example of general device ontology (from [40]).

A number of possible languages can be used; many of them evolved from creation of ontology construction methodologies. The Open Knowledge Base Connectivity (OKBC) [1] model and languages like KIF [2] are examples that have become the bases of other ontology languages. Several languages use frame logic which is basically an object-oriented approach defining frames and attributes (classes and properties). There are also several languages based on description logic, e.g. Loom [3], DAML+OIL [4], or later evolved Web Ontology Language (OWL) [5] standard.

5.1.1.1 Classification of ontology languages

Representation languages can be divided in terms of different abstraction levels used to structure the representation itself:

- Extensional level: basic objects from domain and their relevant properties are described.
- Intensional level: objects are grouped to form concepts, concepts and their properties are specified.
- Meta-level: concepts from intensional level are abstracted, higher level concepts are specified, and previous concepts are seen as instances of new concepts.

Three main questions can be analyzed here:

1. What can be expressed?
 - a) Class and relations: languages aiming at representing objects, classes and relations.
 - b) Actions and processes: languages that provide specialized representation structures for describing dynamic characteristics of the domain, such as actions, processes, and workflows (they usually can represent static aspects of domain too, but only in elementary level).
 - c) Everything: languages that may be used for any kind of contexts and applications.
2. How the context is expressed? (basic formal nature of languages is consider in this criterion)
 - a) Programming languages: allow representation and manipulation of data in several ways and according to various paradigms, leading to a cleaner separation between data structures and algorithms that handle them. Object oriented paradigm is preferred in recent years. This approach is generally associated with a number of concepts, such as complex objects, object identity, methods, encapsulation, typing and inheritance. Example can be language F-logic [6], logical formalism that tries to capture the features of object-oriented approaches to computation and data representation. F-Logic forms the core of systems such as Ontobroker [7].
 - b) Conceptual and semantic database models: semantic (or conceptual) models were introduced as schema design tools. Examples of proposed semantic data models are ER and Extended ER data model, FDM (Functional data model), SDM (Semantic Data Model). Semantic models provide more powerful abstractions for the specification of databases.
 - c) Information system/software formalisms: here belong different formalisms for information system design, especially in object-oriented design. Most widely used formalism is Unified Modelling Language (UML). UML was designed for human-to-human communication of models for building systems in object-oriented programming languages. Over the years its use has been extended to a variety of different aims, including the design of databases schemas, XML document schemas, and knowledge models.
 - d) Logic-based: very important class of languages is based on logic. Such languages express a domain-ontology in terms of the classes of objects that are of interest in the domain, as well as the relevant relationships holding among such classes. These languages have a formal well-defined semantics. Three different types of logic-based languages exist – languages based on first-order predicate logic (e.g. KIF [2]),

languages based on description logics (e.g. OWL [5]), and process-action specification-languages (e.g. PSL [8]).

- e) **Frame-based:** frame is a data structure that provides a representation of an object or a class of objects or a general concept or predicate. Some systems define only a single type of frame, other have two or more types, such as class frames and instance frames. The slots of a frame describe attributes of represented concept. They may also have other components in addition to the slot name, value and value restrictions, for instance the name of a procedure than can be used to compute the value of the slot – facets. Frames are usually organized into taxonomies. Through taxonomic relations, classes may be described as specializations of more generic classes with inheritance capability. Frame-based ontology languages were often used in many knowledge-based applications, like Ontolingua [9], OCML [10], OKBC [11] or XOL [12].
 - f) **Graph-based:** formalisms based on various kinds of graph based or graph-oriented notations. Semantic networks [13] and conceptual graphs [14] originated from the Artificial Intelligence community. OML/CKML (Conceptual Knowledge Markup Language) [15] is a framework and markup language for knowledge and ontology representation based on conceptual graphs. Topic Maps [16] are recent proposal originated from the XML community.
3. **XML-related formalisms:** XML [17] is a tag-based language for describing tree structures with a linear syntax and it is a standard language for exchange of information in the Web. Given the popularity of XML in exchange of information, XML-related languages have been considered as suitable for ontology representation. Important languages are based on Resource Description Framework (RDF) [18]. These provide a foundation for processing metadata about documents.
- a) How the expression can be interpreted? (various languages deal with the representation of incomplete information in different way)
 - b) **Single model:** ontology should be interpreted in such a way that only one model of the corresponding logical theory is a good interpretation of the formal description.
 - c) **Several models:** ontology should be interpreted as specifying what we know about the domain with the reservation that the amount of knowledge we have about the domain can be limited (e.g. first-order logic based languages).

5.1.1.2 Semantic Web Ontology languages

Description Logics (DLs) are a family of logic-based knowledge representation formalisms designed to represent and reason about the knowledge of an application domain in a structured and well-understood way. The basic notions in DLs are concepts and roles, which denote sets of objects and binary relations, respectively. Most of today's semantic web ontology languages are DL-based. Also many of them are XML-related, or they possible XML notation.

Several ontology languages have been designed for use in the web. Among them, the most important are OIL [19], DAML-ONT [20] and DAML+OIL [21]. More recently, a new language, OWL [5], is being developed by the World Wide Web Consortium (W3C) Web Ontology Working Group, which had to maintain as much compatibility as possible with pre-existing languages and is intended to be proposed as the standard Semantic Web ontology language. The idea of the semantic Web is to annotate web pages with machine-interpretable description of their content. In such a context, ontologies are expected to help automated processes to access information, providing structured vocabularies that explicate the relationships between different terms.

Extensible Markup Language

Extensible Markup Language (XML) [17] was widely accepted and used as a convenient information representation and exchange format. XML itself don't carry semantics, but is serves as the base syntax for the leading ontology languages that we shall survey. Later additions like XML-DTD (Document Type Definition) and XML-Schema, added some syntactic rules like enumerations, cardinality constrains, and data types, but still lacked even simple semantics like inheritance.

The purpose of XML Schema is therefore to declare a set of constraints that an XML document has to satisfy in order to be validated. With respect to DTD, however, XML Schema provides a considerable improvement, as the possibility to define much more elaborated constraints on how different part of an XML document fit together, more sophisticated nesting rules, data-typing. Moreover, XML-Schema expresses shared vocabularies and allows machines to carry out rules made by people. Among a large number of other rather complicated features.

Resource Description Framework

Resource Description Framework (EDF) [18] is a standard way for defining of simple descriptions. RDF is for semantics - a clear set of rules for providing simple descriptive information. RDF enforces a strict notation for the representation of information, based on resources and relations between them.

As referred to in its name, RDF strength is in its descriptive capabilities, but is still lacks some important features required in an ontology language such as inferences for example. However, ontology languages built on top of RDF as a representation and description format.

The RDF data model provides three object types: resources, properties, and statements. Resource may be either entire Web page, a part of it, a whole collection of pages or an object that is not directly accessible via the Web, property is a specific aspect, characteristic attribute, or relation used to describe a resource, statement is a triple consisting of two nodes and a connecting edge. These basic elements are all kinds of RDF resources. According to the latter description, a subject is a resource that can be described by some property. For instance a property like name can belong to a dog, cat, book, plant, person, and so on. These can all serve the function of a subject. The predicate defines the type of property that is being attributed. Finally, the object is the value of the property associated with the subject.

RDF Schema

RDF Schema (RDFS) [22] enriches the basic RDF model, by providing a vocabulary for RDF, which is assumed to have certain semantics. Predefined properties can be used to model instance of and subclass of relationships as well as domain restrictions and range restrictions of attributes. Indeed, the RDF schema provides modelling primitives that can be used to capture basic semantics in a domain neutral way. That is, RDFS specifies metadata that is applicable to the entities and their properties in all domains. The metadata then serves as a standard model by which RDF tools can operate on specific domain models, since the RDFS meta-model elements will have a fixed semantics in all domain models.

RDFS provides simple but powerful modelling primitives for structuring domain knowledge into classes and sub classes, properties and sub properties, and can impose restrictions on the domain and range of properties, and defines the semantics of containers.

Web Ontology Language

The next layer in the Semantic Web architecture is Web Ontology Language (OWL) [5], a language for Web ontologies definition and instantiation. OWL enhances RDF vocabulary for describing properties and classes: relations between classes (e.g. subclasses), cardinality, equality, richer typing of properties, characteristics of properties (e.g. symmetry) and instances. OWL is the W3C recommendation for ontology definition, but other standards also support similar characteristics (DAML+OIL). Numerous tools support modelling with OWL and DAML+OIL.

The OWL language also provides three increasingly expressive sublanguages: OWL Lite, OWL DL, and OWL Full, each offers a different level of expressiveness at the trade-off for simplicity, thus offering a suitable sub language parts available for use according to expressibility needs.

There also exists OWL based enhancement to web services oriented languages, aiming to handle semantic descriptions of such services. OWL-S [23] is framework for containing and sharing ontological description of the capabilities and characteristics of a Web service. An OWL-S specification includes three sub-ontologies that define essential types of knowledge about a service – service profile describes the outlining interface and characteristics of the service, a process profile defines the control flow of the service and the service grounding provides mapping with communication-level protocols. OWL-S has similar characteristics with a number of related protocols.

The popularity of OWL-S in the Semantic Web community, as Web services description language, adds to the attractiveness of the language.

5.1.1.3 Ontology Reasoning

We use the term reasoning over ontology to mean any mechanism/procedure for making explicit a set of facts that are implicit in ontology. There are many reasons why one would like to have well-founded methods for reasoning about ontologies. Here, we would like to single out two important purposes of reasoning:

- **Validation.** Validating ontology means ensuring that the ontology is a good representation of the domain of discourse that you aim at modelling. Reasoning is extremely important for validation. For example, checking whether your ontology is internally consistent is crucial: obviously, no inconsistent theory can be a good representation of domain.
- **Analysis.** In analysis one assumes that the ontology is a faithful representation of the domain, and tries to deduce facts about the domain by reasoning over the ontology. In a sense, we are trying to collect new information about the domain by exploiting the ontology. Obviously, analysis can also provide input to the validation phase.

OWL is quite a sophisticated language. The most important feature is capability for DL reasoning, the frames paradigm and the Semantic Web vision of a stack of languages including XML and RDF. On the one hand, OWL semantics is formalised by means of a DL style model theory. In particular, OWL is based on the SH family of Description Logics [24]. Such family of languages represents a suitable balance between expressivity requirements and computational ones. Moreover, practical decision procedures for reasoning on them are available, as well as implemented systems such as RACER [25]. RACER is a Semantic Web inference engine for developing ontologies, answering queries over RDF documents and RDFS/DAML ontologies, registering permanent queries (e.g., for building a document management system) with notification of new results if available (publish-subscribe facility). RACER is a Description Logic reasoning system with support for TBoxes with generalized concept inclusions, ABoxes. Concrete domains (e.g., linear (in-)equalities over the reals), RACER is a theorem prover for modal logic K_m with graded modalities and axioms.

On the other hand, OWL formal specification is given by an abstract syntax that has been heavily influenced by frames and constitutes the surface structure of the language. Class axioms consist of the name of the class being described, a modality indicating whether the definition of the class is partial or complete, and a sequence of property restrictions and names of more general classes, whereas property axioms specify the name of the property and its various features. Such a frame-like syntax makes OWL easier to understand and to use.

Moreover, axioms can be directly translated into DL axioms and they can be easily expressed by means of a set of RDF triples. Given the huge number of requirements for OWL and the difficulty of satisfying all of them in combination, three different versions of OWL have been designed:

- OWL DL, that is characterized by an abstract frame-like syntax and a decidable inference; it is based on SHOIN(D) DL, which extends SH family of languages with inverse roles, nominals (which are, in their simplest form, special concept names that are to be interpreted as singleton sets), unqualified number restrictions and support for simple datatypes; SHOIN(D) is very expressive but also difficult to reason with, since inference problems have NEXPTIME complexity
- OWL Lite, that constitutes a subset of OWL DL that is similar to SHIF(D) and, as such, it does not allow to use nominals and allows only for unqualified number restrictions; inference problems have EXPTIME complexity and all OWL DL descriptions can be captured in OWL Lite, except those containing either individuals names or cardinalities greater than 1.
- OWL Full, that, unlike the OWL DL and OWL Lite, allows classes to be used as individuals and the language constructors to be applied to the language itself; it contains OWL DL but goes beyond it, making reasoning un-decidable; moreover, the abstract syntax becomes inadequate for OWL Full, which needs all the official OWL RDF/XML exchange syntax expressivity.

5.1.2 Ontology evolution

This chapter will describe ontology evolution, terms like ontology management, ontology versioning, or ontology modification will be defined as well as how changes can be captured, represented and propagated in ontology.

5.1.2.1 Basic definitions

According to [26], Ontology Evolution is the timely adaptation of ontology to the arisen changes and the consistent propagation of these changes to dependent artefacts. The author describes ontology evolution as a process, as changes in the ontology can cause inconsistencies in other parts of the ontology, as well as in the dependent artefacts. The ontology evolution process encompasses the set of activities, both technical and managerial, that ensures that the ontology continues to meet organizational objectives and users needs in an efficient and effective way.

It is important to distinguish between the management, modification, evolution and versioning of ontologies. Terminology used in [26] has been adapted from the terminology from the database community [27].

Ontology management is the whole set of methods and techniques that is necessary to efficiently use multiple variants of ontologies from possibly different sources for different tasks. Therefore, an ontology management system should be a framework for creating, modifying, versioning, querying, and storing ontologies. It should allow an application to work with an ontology without worrying about how the ontology is stored and accessed, how queries are processed, etc.;

Ontology modification is accommodated when an ontology management system allows changes to the ontology that is in use, without considering the consistency;

Ontology evolution is accommodated when an ontology management system facilitates the modification of ontology by preserving its consistency;

Ontology versioning is accommodated when an ontology management system allows handling of ontology changes by creating and managing different versions of it.

5.1.2.2 Ontology evolution process and frameworks

As defined in [28] and [29], the authors identify a six-phase evolution process, where the individual phases are:

- Change capturing
- Change representation
- Semantics of change
- Change implementation
- Change propagation
- Change validation

Change capturing

The process of ontology evolution starts with capturing changes either from explicit requirements or from the result of change discovery methods, which induce changes from existing data. Explicit requirements are generated, for example, by ontology engineers who want to adapt the ontology to new requirements or by the end-users who provide the explicit feedback about the usability of ontology entities. The changes resulting from this kind of requirements are called top-down changes. Implicit requirements leading to so-called bottom-up changes are reflected in the behaviour of the system and can be discovered only through the analysis of this behaviour. In [26] author defines three types of change discovery: structure-driven, usage-driven and data-driven. Whereas structure-driven changes can be deduced from the ontology structure itself, usage-driven changes result from the usage patterns created over a period time. Data-driven changes are generated by modifications

to the underlying dataset, such as text documents or a database, representing the knowledge modelled by ontology.

Change Representation

To resolve changes, they have to be identified and represented in a suitable format. That means, the change representation needs to be defined for a given ontology model. Changed can be represented on various levels of granularity, e.g. as elementary or complex changes. A common practice is to provide a taxonomy or ontology of changes for a given ontology model.

Semantics of Change

The semantics of change refers to the effects of the change on the ontology itself, and, in particular the checking and maintenance of the ontology consistency after the change application. The meaning of consistency very much depends on the underlying ontology model. It can for example be defined using a set of constraints, as in the KAON ontology model, or it can be given a model-theoretic definition.

Change Propagation

Ontologies often reuse and extend other ontologies. Therefore, an ontology update might also corrupt ontologies depending on the modified ontology (through the inclusion, mapping integration, etc.) and consequently, all the artefacts based on these ontologies. The task of the change propagation phase of the ontology evolution process is to ensure consistency of dependent artefacts after an ontology update has been performed. These artefacts may include dependent ontologies, instances, as well as application programs running against the ontology.

Change Implementation

The role of the change implementation phase of the ontology evolution process is:

- To inform an ontology engineer about all consequences of a change request
- To apply all the (required and derived) changes and
- To keep track about performed changes.

Change Validation

There are numerous circumstances where it may be desired to reverse the effects of the ontology evolution, to name just a few:

- The ontology engineer may fail to understand the actual effect of the change and approve the change that should not be performed;
- It may be desired to change the ontology for experimental purposes;
- When working on ontology collaboratively, different ontology engineers may have different ideas about how the ontology should be changed.

It is the task of the change validation phase to recover from these situations. Change validation enables justification of performed changes and undoing them at user's request. Consequently, the usability of the ontology evolution system is increased.

5.1.3 Ontology versioning

Ontology versioning is a stronger variant of handling changes to ontologies: While ontology evolution is concerned about the ability to change ontology without losing data and by maintaining consistency, ontology versioning allows accessing the data through different variants of the ontology. In addition to managing the individual variants of the ontology themselves, it is also important to manage the derivation relations between the variants. These derivation relations then allow defining the notions of compatibility between versions, mapping relations between versions, as well as transformations of data corresponding to the various versions.

The problem of schema evolution has been extensively studied especially in the context of object-oriented databases. Dynamic schema evolution in databases is defined as managing schema changes in a timely manner without loss of existing data while the database system continues to be operational and without significantly impacting day-to-day operations of the database. Particular

problems addressed are cascading changes (changes required to other parts of the schema as a result of a change), ensuring consistency of the schema, and propagation of the changes to the corresponding database. Although there are significant differences between schema evolution and ontology evolution [30], many of the methods and technologies developed for schema evolution can be applied or adapted to ontology evolution.

The problem of evolution and versioning is also present in other application areas of information systems. For example, Concurrent Versions Systems (CVS) allow the concurrent update to files while maintaining the version history of those files as well as detecting and resolving conflicts in updates to the files. Another application is the maintenance of knowledge-base systems and belief revision, where knowledge base (e.g. believes of an agent) needs to be update to incorporate new information while maintaining consistency of the knowledge base.

5.2 Change management

5.2.1 Content syndication

Content syndication is a general term used to refer to accessing and publishing web content (text, images, etc.). Content publishers can make their content available through syndication by using available technologies to produce what is known as 'feeds' (e.g. 'blog feeds' or 'news feeds'). These feeds can either show headlines only, headlines and summary or full content. The focus is mainly on dynamic content that allows people to share information and to interact. In general, content syndication refers to making feeds available from a site in order to provide other people an updated list of content from it (for example one's latest forum postings, etc.). This originated with news and blog sites but is increasingly used to syndicate any information. Anything that can be broken down into discrete items can be syndicated: the "recent changes" of a wiki, a changelog of CVS checkings, even the revision history of a book, etc.

Readers or fellow web publishers can access the content (e.g. latest updates) of particular sites with content syndication when they use aggregators or feeds generators. Once information about some item(s) is in appropriate format, a feed-aware program can check the feed for changes and react to the changes in an appropriate way.

Basic mechanism consists of four steps to publishing and receiving a message:

1. **Step 1:** The publisher creates content and publishes (in the form of an XML file). This file is called the feed, which is the container into which messages are sent. The XML file has a URL associated with it, just like any other Web page. The publisher can then post that URL on his or her Web site.
2. **Step 2:** The recipient who wants to receive the feed adds that URL into a program called a reader or aggregator. A reader is just a program located either on the Web or on the recipient's desktop that can read and interpret the particular XML feed files.
3. **Step 3:** When the publisher wants to send a message to the recipient, he or she simply adds entries to the XML file. Messages have three parts: a title, a summary and a message body. The title and the summary are added directly to the XML file, along with an entry date. The message body is an HTML file that is referenced in the entry.
4. **Step 4:** The function of the reader is to make "virtual visits" to the specified XML Web page at specific intervals (usually once an hour) and check for updates. When the reader finds updates, it makes them available to the recipient. The form in which the message is delivered depends on the reader. For example, it might appear as an entry under the feed banner or it would look like an email message.

Currently, two leading technologies heavily used are RSS and Atom. Both of them are XML based formats (files must conform to the XML 1.0 specification, as published on the World Wide Web Consortium website.) with elements enabling to describe channels and items within these channels. To describe channels and items in more detail various tags for different bits are used e.g. title, link, description, etc. In addition to marking basic information, each format enables to mark some

additional information. Statistics on using them shows that currently 78 % of feeds are based on RSS and 18 % on Atom (but it is necessary to keep in mind that Atom is younger than RSS) [31].

5.2.1.1 Standards for content syndication

RSS

The name "RSS" is an umbrella term for a format that spans several different versions. The original RSS, version 0.90, was designed by Netscape as a format for building portals of headlines to mainstream news sites. It was deemed overly complex for its goals; a simpler version, 0.91, was proposed and subsequently dropped when Netscape lost interest in the portal-making business. But 0.91 was picked up by another vendor, UserLand Software, which intended to use it as the basis of its weblogging products and other web-based writing software.

In the meantime, a third, non-commercial group (RSS-DEV Working Group around the W3C's RDF standard) designed a new format based on RSS 0.90 (before it got simplified into 0.91). This format, which is based on RDF, is called RSS 1.0 (released in 2000). But UserLand was not involved in designing this new format, and, as an advocate of simplifying 0.90, it was not happy when RSS 1.0 was announced. Instead of accepting RSS 1.0, UserLand continued to evolve the 0.9x branch, through versions 0.92, 0.93, 0.94, and finally 2.0 (released and declared frozen in 2002).

Therefore, two versions are available currently (older versions, although some of them still quite popular among developers and still in use, are considered obsolete since they are superseded by one of the current versions) [32], [33]:

- RSS 1.0 (RDF Site Summary): recommended use for RDF-based applications or if there is a need for advanced RDF-specific modules
- RSS 2.0 (Really Simple Syndication): recommended use for general-purpose, enabling metadata-rich syndication

RSS 1.0 unlike RSS 2.0 has the form of an XML serialisation of RDF document and can be characterised by an extensive use of namespaces (also default namespace is defined). It uses Dublin Core for the additional metadata of article authors and publishing dates. RSS 1.0, due to its RDF based nature, offers a variety of benefits. However, uptake for RSS 1.0 has been relatively limited, due to the difficulty in creating conforming documents in comparison to other syndication formats. An attempt to update RSS 1.0 to RSS 1.1 [34] appeared without any visible success.

According to statistics presented on [31], RSS formats are distributed as follows: RSS 2.0 - 68 %, RSS 1.0 – 17 %, RSS 0.91 – 12 %. RSS 2.0 and RSS 0.91 are compatible as they are from the same company. On the other hand, RSS 1.0 and 2.0 are not compatible.

Atom

Atom Syndication Format is an open standard. The development of Atom was motivated by perceived deficiencies in the RSS 2.0 format. Community was unable to make changes directly to RSS 2.0, because it was not an open standard. RSS 2.0 was copyrighted by Harvard University and in the official specification document it stated that it was purposely frozen.

In June 2003 a project was initiated, aims of which were to develop a web syndication format that was:

- 100% vendor neutral,
- implemented by everybody,
- freely extensible by anybody, and
- cleanly and thoroughly specified.

A project snapshot known as Atom 0.2 was released in early July 2003. The next snapshot was Atom 0.3, released in December 2003. This version gained widespread adoption in syndication tools, and in particular it was added to several Google-related services, such as Blogger, Google News and Gmail. The final draft of Atom 1.0 was published in July 2005 and was accepted by the IETF (Internet Engineering Task Force) as a "proposed standard" in August of 2005. In December 2005,

The Atom Syndication Format was issued as a proposed "internet official protocol standard" in IETF RFC 4287 [35].

Standard comparison

When we talk about content syndication, there are a few different aspects we need to talk about. But the most interesting facet is the "feed format", which is really the schema of the XML that represents the channel and items within that channel. It is necessary to say that these formats are way more alike than they are different.

In fact, for basic functionality except for some special issues, it is possible to do the transcoding (see the section on software converters). They all deal with representing a feed (called a <channel> in RSS 1.0 and RSS 2.0, a <feed> in Atom) of content items (called an <item> in RSS 1.0 and RSS 2.0, an <entry> in Atom) with certain (and slightly different) attributes of the feed and items being considered "core". There are no show stoppers translating from one feed to another, just annoyances.

Very important thing is that it is possible to extend all three formats in a well-defined manner using what are called "modules" or "namespaces" ("modules" that use "XML namespaces"). A properly defined module can be referenced in all three formats. This is the important part! So, for example, while RSS 1.0 relies on using the Dublin Core to represent much of the metadata associated with an item, there's no reason that an RSS 2.0 or Atom feed can't reference the Dublin Core module and use the exact same elements.

There is no clear winner – the best way is to prefer different feed formats for different reasons: RSS 2.0 is simple, but many developers do not like the date format it uses or the looseness of the specification. Others like Atom's model for representing content, but do not like the fact it is early in the adoption curve (with implication on availability of tools and libraries). Some people may like RSS 1.0 for its formality while the others do not like it for its formality – in fact RDF is good, but still some quite significant work is needed.

More in-depth comparison is in [36] (comparing RSS 2.0 and Atom) and in [37] (comparing different versions of RSS).

5.2.1.2 Software tools

Aggregators/readers

Aggregator is a program/tool, which enables individuals to read content (or site content updates) using content feeds. Aggregators are set up to periodically check for new items in the feeds someone is subscribed to. In other words, the news comes to him, rather than he having to go to the news.

There are two main types of aggregators: web-based aggregators and desktop/software aggregators. Web-based aggregators allow individuals to sign up for the service and read their feeds online in just one site. There's no need to download and install any programs Desktop/software aggregators require individuals to download and install a program to the computer. This type of aggregator usually has a lot more functions available to the user.

Validators

Despite its relatively simple nature, content feeds can (and very often are) poorly implemented by many tools. Feed validators represent an attempt to codify the specification to make it easier to know when someone is producing content feed correctly, and to help him fix it when he is not.

Feed generators

Feed generator is a type of software tool which allows users to easily create, edit and publish RSS feeds. Day to day feed maintenance can be performed.

Feed converters

Since there are several different formats in use, user can face a case his/her preferred feed reader is not compatible with a feed time for conversion between different formats.

5.2.1.3 Content syndication in knowledge management

Content syndication really only covers delivery of content items; it doesn't deal with storage of stuff or keeping track of relationships or anything like that. RSS/Atom can represent a solution to many problems both on the input side as well as on the output side. On the input side enables to integrate many (hundreds) feeds and then skim them all at once, with much less effort than having to visit all those content resources individually. It is good on the output side, too - it gives a nice smooth way of getting information (hints, alerts, required information, etc.) out to people who want to read it. Therefore, many solutions (both commercial and open sourced) for knowledge management try to integrate syndication technology seamlessly into its interfaces.

5.2.1.4 Individualised feeds

Individualised RSS (IRSS) is an evolution of the RSS standard which will allow content to be published in a way that can be targeted, measured and individualised [38]. This means that every subscriber to an RSS feed can receive unique content meant only to him or her or a specific group. It allows for fully individualised communications such as alerts, notices and targeted promotions.

Individualised RSS feeds allow content providers to target, segment and individualise communications much the way they do email messages today. Individualised RSS recipients receive text, images and other bits of content uniquely matched to their expressed interests and desires. The individualised feeds enable providers to communicate with subscribers based on demographics, past behaviour, or any other segmenting attributes.

With these solutions, each recipient gets his or her own unique feed, enabling providers to understand exactly how many and which recipients are picking up their messages. And because each feed is unique to the individual recipient, providers can track and measure subscriber actions all the way down to an individual, facilitating the same behavioural targeting and testing possible in other individualised media. Moreover, providers can actually create a unique message for each user based upon demographic or behavioural data.

But best of all, these individualised RSS solutions do not require any changes on the part of recipients – they can use the same reader they use today. Three types of IRSS can be identified:

- **Metric Enabled** (using unique URLs to identify unique users, but their content and structure are always the same)
- **Customised Feeds** (carry different content items for different users. The content items themselves are the same, but different users will get different items)
- **Personalised Feeds** (content items may differ, for example by including the name of the recipient and data unique to that recipient)

Metric Enabled RSS feeds are using unique URLs to identify unique users, but their content and structure are always the same. The solution to this problem is adding some additional meta data to the RSS specification, which would allow the aggregators to cache the feed, but still enable the metrics.

Customised feeds carry different content items for different users. The content items themselves are the same, but different users will get different items. The solution to this problem is adding additional meta data to the content item itself, to let the aggregators identify individual content items, regardless of what feed from a certain publisher they appear in.

In personalised feeds the actual content items may differ, for example by including the name of the recipient and data unique to that recipient. This can again be solved by meta-data, which would tell the aggregators that this content item in fact is unique.

5.2.1.5 Attention.XML

There is one problem with using content syndication technologies in practice - feed readers collect updates, but with too many unread items, how do you know which to read first? Attention.XML [39] is designed to solve these problems and enable a whole new class of blog and feed related

applications. It is an open standard, built on open source that helps keep track of what people have read, what they are spending time on, and what they should be paying attention to.

Attention.XML is an XML file that contains an outline of feeds/blogs, where each feed itself is an outline, and each post is also an outline under the feed. This hierarchical outline structure is then annotated with per-feed and per-post information which captures such information as, the last time the feed/post was accessed, the duration of time spent on the feed/post, recent times of feed/post access, user set (dis)approval of posts, etc.

Basically it is metadata that records and shares information on the "attention" users give to their RSS feeds and blogs. Attention.xml basically provides a way of describing aspects of a user's visits to a blog/feed/page/post/item/entry in a machine-readable fashion. This is information that could be extremely useful if captured, to both clients and servers of feeds. Attention.xml could tell us who looks at a blog or feed, how often they look at it, where those viewers come from.

5.2.2 Version control

Version (Revision) control is the management of multiple revisions of the same unit of information. It is most commonly used in engineering and software development to manage ongoing evolution of digital documents like application source code, art resources such as blueprints or electronic models and other critical information that may be worked on by a team of people. Changes to these documents are identified by incrementing an associated number or letter code, termed the "revision number", "revision level", or simply "revision" and associated historically with the person making the change. A simple form of revision control, for example, has the initial issue of a drawing assigned the revision number "1". When the first change is made, the revision number is incremented to "2" and so on.

Software tools for revision control are increasingly recognized as being necessary for most software development projects.

5.2.2.1 Overview

Engineering revision control developed from formalized processes based on tracking revisions of early blueprints or blueines. Implicit in this control was the option to be able to return to any earlier state of the design, for cases in which engineering dead-end was reached in iterating any particular engineering design. Likewise, in computer software engineering, revision control is any practice which tracks and provides controls over changes to source code. Software developers sometimes use revision control software to maintain documentation and configuration files as well as source code. In theory, revision control can be applied to any type of information record. However, in practice, the more sophisticated techniques and tools for revision control have rarely been used outside software development circles (though they could actually be of benefit in many other areas). However, they are beginning to be used for the electronic tracking of changes to CAD files, supplanting the "manual" electronic implementation of traditional revision control.

As software is developed and deployed, it is extremely common for multiple versions of the same software to be deployed in different sites, and for the software's developers to be working privately on updates. Bugs and other issues with software are often only present in certain versions (because of the fixing of some problems and the introduction of others as the program evolves). Therefore, for the purposes of locating and fixing bugs, it is vitally important for the debugger to be able to retrieve and run different versions of the software to determine in which version(s) the problem occurs. It may also be necessary to develop two versions of the software concurrently (for instance, where one version has bugs fixed, but no new features, while the other version is where new features are worked on).

At the simplest level, developers can simply retain multiple copies of the different versions of the program, and number them appropriately. This simple approach has been used on many large software projects. Whilst this method can work, it is inefficient (as many near-identical copies of the program will be kept around), requires a lot of self-discipline on the part of developers, and often

leads to mistakes. Consequently, systems to automate some or all of the revision control process have been developed.

Traditionally, revision control systems have used a centralized model, where all the revision control functions are performed on a shared server. A few years ago, systems like TeamWare, BitKeeper, and GNU arch began using a distributed model, where each developer works directly with their own local repository, and changes are shared between repositories as a separate step. This mode of operation allows developers to work without a network connection, and it also allows developers full revision control capabilities without requiring permissions to be granted by a central authority. One of the leading proponents of distributed revision control is Linus Torvalds, developer of the Linux kernel.

In most software development projects, multiple developers work on the program at the same time. If two developers try to change the same file at the same time, without some method of managing access the developers may well end up overwriting each other's work. Most revision control systems solve this in one of two ways. This is only a problem for centralized revision control systems, since distributed systems inherently allow multiple simultaneous editing.

Some systems prevent "concurrent access" problems, by simply locking files so that only one developer at a time has write access to the central "repository" copies of those files. Others, such as CVS, allow multiple developers to be editing the same file at the same time, and provide facilities to merge changes later. In the latter type, the concept of a reserved edit can provide an optional means to explicitly lock a file for exclusive write access, even though a merging capability exists.

The merits and drawbacks of file locking are hotly debated. It can provide some protection against difficult merge conflicts when a user is making radical changes to many sections of a large file (or group of files). But if the files are left exclusively locked for too long, other developers can be tempted to simply bypass the revision control software and change the files locally anyway. That can lead to more serious problems.

Some systems attempt to manage who is allowed to make changes to different aspects of the program, for instance, allowing changes to a file to be checked by a designated reviewer before being added.

Most revision control software use delta compression, which retains only the differences between successive versions of files. This allows more efficient storage of many different versions of files.

Some of the more advanced revision control tools offer many other facilities, allowing deeper integration with other tools and software engineering processes. Plugins are often available for IDEs such as Eclipse and Visual Studio, NetBeans IDE comes with integrated version control support.

The Wikipedia:Page history features of MediaWiki are identical in concept and practice to the revision control software discussed above.

5.2.2.2 Basic terms

Repository is where the file data is stored, often on a server. Working copy is the local copy of files from a repository, at a specific time or revision. All work done to the files in a repository is done on a working copy, hence the name. Check-Out creates a local working copy from the repository. Either a revision is specified, or the latest is used. Commit occurs when a copy of the changes made to the working copy is made to the repository.

Change in revision control represents a specific modification to a document under version control. The granularity of the modification considered a change varies between version control systems. Then change List identifies the set of changes made in a single commit. This can also represent a sequential view on the source code, allowing source to be examined as of any particular changelist ID.

Update is a process of copying of changes that were made to the repository into the local working directory. A merge or integration brings together (merges) concurrent changes into a unified revision. From this point of view revision (or version) is one version in a chain of changes.

Import is the action of copying a local directory tree (not a working copy) into the repository. Export is similar to a check-out except that it creates a clean directory tree without the version control metadata used in a working copy. Often used prior to publishing the contents. Conflict occurs when two changes are made by different parties to the same document or place within a document. Since the software may not be intelligent enough to decide which change is 'correct', a user is required to resolve the conflict. Resolve is act of user intervention to address a conflict between different changes to the same document. Baseline is approved revision of a document or source file from which subsequent changes can be made.

5.2.3 Ontology-based change management

An important task after creating ontology is its storing and maintenance. These tasks are referred to as ontology management. In practice, ontologies are not static, but evolve over time. A support to handle this evolution is needed. This is especially important when ontologies are used in a decentralized and uncontrolled environment like the Web, where changes occur without coordination, where this may have unexpected and unknown results. There are several reasons for changes in ontologies. Changes in ontologies can be caused by:

- changes in the domain;
- changes in the conceptualization;
- changes in the specification.

Storage is the problem of ontology library systems. These systems fall into one of the two categories:

- those with a client/server-based architecture aimed at enabling remote accessing and collaborative editing (WebOnto, Ontolingua, DAML Ontology Library);
- those that feature a web-accessible architecture (SHOE, IEEE SUO).

Ontology Server features a database-structured architecture. Most ontologies are classified or indexed. They are stored in a modular structured library (or lattice of ontologies). WebOnto, Ontolingua and ONIONS all highlight the importance of a modular structure in an ontology library system as that structure facilitates the task of re-organizing ontology library systems and re-using and managing ontologies.

5.3 References

- [1] Corcho, O., Fernandez-Lopez, M., Gomez-Perez, A., "Methodologies, tools and languages for building ontologies. Where is the meeting point?" *Data&Knowledge Engineering* 46, pp. 41-64, 2003.
- [2] Genesereth, M.R., Fikes, R.E., "Knowledge Interchange Format. Version 3.0. Reference Manual." Stanford University, 1992.
- [3] MacGregor, R., Bates, R., "The Loom Knowledge Representation Language". Technical Report ISI-RS-87-188, USC Information Sciences Institute, Marina del Rey, CA, 1987.
- [4] Connolly, D., van Harmelen, F., Horrocks, I., McGuinness, D., Patel-Schneider, P.F., Stein, L.A., DAML+OIL (March 2001) Reference Description. W3C Note, 2001.
- [5] McGuinness, D., van Harmelen, F., "OWL Web Ontology Language Overview". W3C Recommendation, 2004.
- [6] Kifer, M., Lausen, G., F-Logic: "A Higher-Order language for Reasoning about Objects, Inheritance, and Scheme", in *Proc. of SIGMOD Conference 1989*, pp. 134-146, 1989.
- [7] Fensel, D., Decker, S., Erdmann, M., Studer, R., Ontobroker: "The Very High Idea", in *Proceedings of the 11th International Flairs Conference (FLAIRS-98)*, Florida, 1998.
- [8] Knutilla, A., et. al., "Process Specification Language: Analysis of Existing Representations", NISTIR 6133, National Institute of Standards and Technology, Gaithersburg, MD, 1998.

-
- [9] Gruber, T.R., A Translation Approach to Portable Ontology Specifications, in Knowledge Acquisition, 5(2), 1993.
- [10] Motta E., Reusable Components for Knowledge Modelling. IOS Press, Amsterdam, Netherlands, 1999
- [11] Chaudhri, V., Farquhar, A., Fikes, R., Karp, P., Rice, J. (1998), OKBC: A programming foundation for knowledge base interoperability, in Proceedings of AAAI'98, pp. 600-607, 1998.
- [12] Karp, R., Chaudhri, V., Thomere, J., XOL: An XML-Based Ontology Exchange Language, Technical report, 1999.
- [13] Sowa, J.F., Semantic networks, available online <http://www.jfsowa.com/pubs/semnet.htm>, 2002.
- [14] Sowa, J.F., Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing Co., Pacific Grove, CA, 2000.
- [15] R.E. Kent., Conceptual Knowledge Markup Language: The Central Core', in: Twelfth Workshop on Knowledge Acquisition, Modeling and Management, 1999.
- [16] TopicMaps.Org Authoring Group, XML Topic Maps (XTM) 1.0 TopicMaps.Org specification, 2001
- [17] Yergeau, F., Bray, T., Paoli, J., Sperberg-McQueen, J.M., Maler, E., Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation, 2004.
- [18] Manola, F., Miller, E., McBride, B., RDF Primer. W3C Recommendation, 2004.
- [19] Fensel, D., Horrocks, I., van Harmelen, F., McGuinness, D., Patel-Schneider, P.F., OIL: Ontology Infrastructure to Enable the Semantic Web. IEEE Intelligent Systems, 16 (2), 2001.
- [20] Stein, L.A., Connolly, D., McGuinness, D., Annotated DAML Ontology Markup. Initial draft, <http://www.daml.org/2000/10/daml-walkthru>, 2000.
- [21] Connolly, D., van Harmelen, F., Horrocks, I., McGuinness, D., Patel-Schneider, P.F., Stein, L.A., DAML+OIL (March 2001) Reference Description. W3C Note, 2001.
- [22] Brickley, D., Guha, R.V., McBride, B., RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, 2004.
- [23] Martin, D., et al., Bringing Semantics to Web Services: The OWL-S Approach, in Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), July 6-9, 2004, San Diego, California, USA.
- [24] Horrocks, I., Sattler, U., Ontology reasoning in the SHOQ(D) description logic. In Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001), pp. 199-204, 2001.
- [25] Haarslev, V., Moller, R., Racer: An owl reasoning agent for the semantic web, in Proc. of the International Workshop on Applications, Products and Services of Web-based Support Systems, in conjunction with 2003 IEEE/WIC International Conference on Web Intelligence, Halifax Canada, Oct 13, pp. 91-95, 2003.
- [26] Stojanovic, L., Methods and Tools for Ontology Evolution, PhD thesis, University of Karlsruhe, 2004.
- [27] Roddick, J.F., A survey of schema versioning issues for database systems. Information and Software Technology, 37(7): pp. 383-393, 1995.
- [28] Haase, P., Sure, Y., State-of-the-Art on Ontology Evolution, SEKT Project IST-2003-506826, Deliverable D3.1.1b, 2004.
- [29] Stojanovic, L., Maedche, A., Motik, B., Stojanovic, N., User-driven ontology evolution management, in European Conf. Knowledge Eng. And Management (EKAW 2002), pp. 285-300. Springer-Verlag, 2002.
- [30] Noy, N.F., Klein, M., Ontology evolution: Not the same as schema evolution. Knowledge and Information Systems, 5, 2003.
- [31] Barr, J., Kearney, B., Syndic8.com – a directory of more than 471 000 (May 2006) feeds, <http://www.syndic8.com/>.

- [32] RSS-DEV Working Group, RSS 1.0 Specification, <http://web.resource.org/rss/1.0/spec>, 2001.
- [33] RSS Advisory Board, RSS 2.0 Specification, <http://www.rssboard.org/rss-specification>, 2005.
- [34] Palmer, S.B., Schmidt, C., RSS 1.1 Specification, <http://inamidst.com/rss1.1/>, 2005.
- [35] Network Working Group, The Atom Syndication Format, available online <http://tools.ietf.org/html/4287>, 2005.
- [36] RSS 2.0 and Atom 1.0, Compared, <http://www.tbray.org/atom/RSS-and-Atom>.
- [37] RSS.NET, RSS Version Comparison, http://www.rssdotnet.com/documents/version_comparison.html, 2006.
- [38] <http://www.razorshine.com/archive/2005/10/26/irss-the-evolution-of-rss/>
- [39] Technorati Developers Wiki, Attention XML, <http://developers.technorati.com/wiki/attentionxml>, 2006.
- [40] Bandara, A., Payne, T. R., de Roure, D. and Clemo, G. An Ontological Framework for Semantic Description of Devices. In Proceedings of International Semantic Web Conference (ISWC), Hiroshima, Japan, 2004.

6 Service-oriented architecture

A Service-Oriented Architecture (SOA) [1] is a technology approach that defines the use of services available on the network (such as www) to support the requirements of software users. In an SOA environment, resources on a network are made available as independent services that can be accessed without knowledge of their underlying platform implementation. Service-oriented architecture is usually implemented as an available set of services, which can be used by other applications, or as the application that uses services (or both). SOA is usually based on Web services standards (e.g., using SOAP [3] or REST [4]) that are widely accepted by industry. These standards (also referred to as Web service specifications) also provide greater interoperability and some protection from lock-in to proprietary vendor software. However, one can implement SOA using any service-based technology.

SOA can support integration and consolidation activities within complex enterprise systems, but SOA does not specify or provide a methodology or framework for documenting capabilities or services.

6.1 Key elements of SOA

6.1.1 Service

A service represents a specific (typically business) function, or it can perform a set of related functions. Multiple services can be used in some coordinated way. The composite or aggregated service can be used for satisfy the more complex requirement. The SOA is an approach to connecting applications that are realised as services, which can communicate to each other. In this context, SOA is a way of sharing functions in a flexible way.

Service oriented architectures have been used for years, so, the SOA concept is not new. What distinguishes the SOA from other similar approaches is so-called loose coupling. All services are abstracted from the internal design that achieves the results for the services. The interface should have sufficient information for a service to be identified and used without needing to know about its internal design, language, or platform implementation. A loosely-coupled design also means that services are designed for no particular service consumer. The information carried by the service should be agnostic to the purpose and technical objectives of the service consumer.

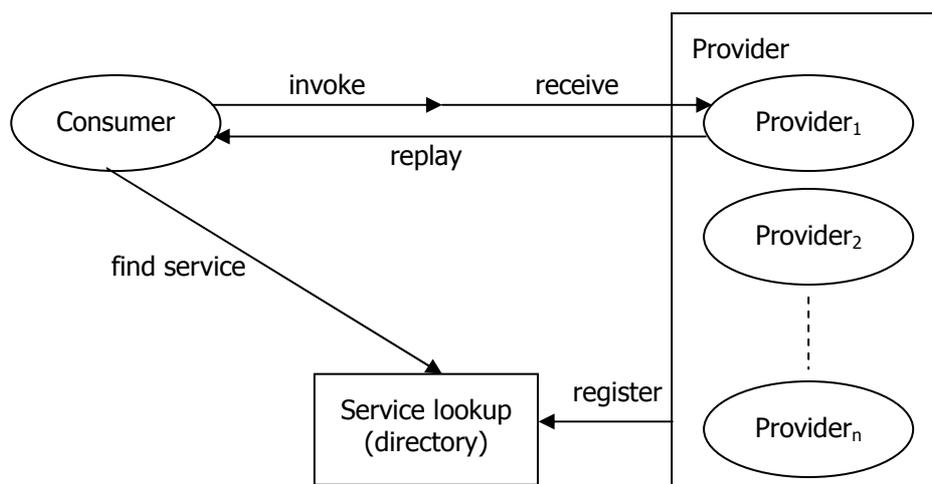


Figure 3 SOA approach

In according to [2], the SOA is defined as a set of components which can be invoked, and whose interface descriptions can be published and discovered. This definition specifies SOA as the group of services, that work together and the cooperation is based on the common semantics. These services are modularly implemented components that can be invoked by a consumer or a client. Their interface can be identified and used with no need for knowledge of its internal design. The producer

and the consumer of these services can be separated from one another, and the services can be registered so that a consumer or client can locate them either statically or dynamically in the registry. The general view of a SOA approach is illustrated at Figure 3.

In a SOA, the location of the service(s) should be transparent to the consumer. The registry could serve as the discovery mechanism for consumers to locate services being offered in a transparent way. An architectural level, it is irrelevant whether the services are local or remote. It is the responsibility of the system, not the calling application, to effect and manage the invocation of the service. This allows for the services to be truly independent and managed.

6.1.2 Messages

Service providers and consumers communicate via messages. Services expose an interface contract. This contract defines the behaviour of the service and the messages they accept and return. Because the interface contract is platform- and language-independent, the technology used to define messages must also be agnostic to any specific platform/language. Therefore, messages are typically constructed using XML documents that conform to XML schema. XML provides all of the functionality, granularity, and scalability required by messages. That is, for consumers and providers to effectively communicate, they need a non-restrictive type of system to clearly define messages; XML provides this. Because consumers and providers communicate via messages, the structure and design of messages should not be taken lightly. Messages need to be implemented using a technology that supports the scalability requirements of services. Having to redesign messages will break the interface to providers, which can prove to be costly.

6.1.3 Dynamic discovery

Dynamic discovery is an important piece of SOA. At a high level, SOA is composed of three core pieces: service providers, service consumers, and the directory service. The role of providers and consumers are apparent, but the role of the directory service needs some explanation. The directory service is an intermediary between providers and consumers. Providers register with the directory service and consumers query the directory service to find service providers. Most directory services typically organize services based on criteria and categorize them. Consumers can then use the directory services' search capabilities to find providers. Embedding a directory service within SOA accomplishes the following:

- Scalability of services; you can add services incrementally.
- Decouples consumers from providers.
- Allows for hot updates of services.
- Provides a look-up service for consumers.
- Allows consumers to choose between providers at runtime rather than hard-coding a single provider.

6.2 Service-orientation principles

In accordance to [3], the service-oriented approach includes the following basic principles:

1. Service reusability: logic is divided into services with the intention of potential reuse.
2. Service contract: services adhere to a communications agreement, as defined collectively by one or more service description documents.
3. Service loose coupling: services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other.
4. Service abstraction: beyond what is described in the service contract, services hide logic from the outside world.

5. Service composability: collections of services can be coordinated and assembled to form composite services.
6. Service autonomy: services have control over the logic they encapsulate.
7. Service statelessness: services minimize retaining information specific to an activity.
8. Service discoverability: services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms.

6.2.1 Service reusability

One of the basic benefits of SOA is the reusability of services. Reusability enables to take the service developed for existing business applications, expose it as services, and then reuse it to meet new business requirements. Reusing functionality that already exists outside or inside an enterprise instead of developing code that reproduces those functions can result in a huge savings in application development cost and time. The benefit of reuse grows dramatically as more and more business services get built, and incorporated into different applications.

In a SOA, the only characteristic of a service that a requesting application needs to know about is the public interface. The functions of an application or system can be dramatically easier to access as a service in an SOA than in some other architecture. So integrating applications and systems can be much simpler.

6.2.2 Service contract

Service contract includes a formal definition of:

- the service endpoint,
- each service operation,
- every input and output message supported by each operation,
- rules and characteristics of the service and its operations.

Service contract may also contain the semantic information that specifies, how some particular task will be realised by the service. The service contract represents the agreement established between providers and consumers of the service.

6.2.3 Service loose-coupling

Loose coupling is the property, which means, that a service can acquire some knowledge from another service, remaining independent from that service. Services are able to interact according to parameters defined in the service contracts. Service contracts are descriptions of the services.

Because services in SOA are loosely coupled, applications that use these services tend to scale easily, certainly more easily than applications in a more tightly-coupled environment. That's because there are few dependencies between the requesting application and the services it uses. The dependencies between client and service in a tightly-coupled environment are compounded (and the development effort made significantly more complex) as an application that uses these services scales up to handle more users.

6.2.4 Service abstraction

The service abstraction is the condition, which means, that a service acts as a black box. The implementation, logic, internal states, states transitions, etc. are hidden from the outside world. The service only exposes the behaviours defined in its contract.

Generally, there are no restrictions on the service capabilities. A service can be designed to perform a simple task, but it may also represent the gateway to a complex solution including the responses

from multiple different systems. Service exposes the functionality represented by the individual operations. Services simply behave as the containers for operations.

6.2.5 Service composability

Generally, a service represents a specific functionality including responses from other services, or that is a part of another services. The principle of composability means, that services should be designed so that they can participate as a functional parts of another service compositions, when required.

A common SOA extension of service composability is so-called concept of orchestrations. In this case, a service-oriented process is controlled by a parent process service that composes process participants. Orchestration defines the control and data flow between services to achieve a business process. Orchestration defines an "executable process" or the rules for a business process flow which can be given to a business process engine to "orchestrate" the process, from the viewpoint of one participant.

The important requirement for a service, to be composable, is the design of service operations. Composability can be viewed as a special form of reuse, so the operations should be designed in a standard way with appropriate level of granularity.

6.2.6 Service autonomy

Service autonomy means, that the functionality of each service exists within an explicit boundary. Each service has its own life and life cycle that is independent of other services. Individual services may be stopped, restarted, and replaced without stopping the whole system. A service can join a process, when a connection is available, and then disconnect later without causing any problem. An obsolete service can be replaced with a new version without downtime. Service autonomy is one a primary considerations when designing how application logic should be divided into services, and which operations should be included within a service context.

The autonomy does not necessarily grant a service exclusive ownership of the functionality it encapsulates. It only guarantees that at the time of execution, the service has control over all functionality it represents. Service boundaries are distinct from each other, but the service can share a specific underlying resources. The underlying logic is under complete control of the service.

6.2.7 Service statelessness

Services should minimize the amount of state information and the time for which they use it. State information is data specific to the current activity of the service. While processing a specific request, a service is usually temporary stateful. If a service needs to hold the state for longer periods of time, its availability for other requestors can be difficult.

In the case of stateless service, each message that a consumer sends to a provider contains all necessary information for the service to process it. This constraint makes a service provider more scalable because the provider does not have to store state information between requests. In the processing cycle, there are no intermediate states, so recovery from partial failure is also relatively easy. This makes a service more reliable.

Stateful service is difficult to avoid in a number of situations. One situation is to establish a session between a consumer and a provider. A session is typically established for efficiency reasons. Another situation is to provide customized service. Stateful services require both the consumer and the provider to share the same consumer-specific context, which is either included in or referenced by messages exchanged between the provider and the consumer. The drawback of this constraint is that it may reduce the overall scalability of the service provider because it may need to remember the shared context for each consumer. It also increases the coupling between a service provider and a consumer and makes switching service providers more difficult.

Statelessness is a preferred condition for services and enables easier scalability and reusability. Statelessness is primarily supported by SOA approach by using document-style messages. The more

information and intelligence added to the messages, the more independent and self-sufficient the service remains.

6.2.8 Service discoverability

Service discovery represents the mechanism for avoiding the creation of redundant services or services implementing redundant logic. Therefore, information (metadata) about the service should include the definition of basic service functionality, but also the specifications of the operations provided by the service.

On a SOA level, discoverability refers to the ability of a architecture to provide a discovery mechanism such as a service registry or directory. This mechanism can support many implementations of SOA. On a service level, the principle of discoverability refers to the design of an individual service so that it will be as discoverable as possible [3].

6.3 Web services

Web service technology is common implementation of Service-Oriented architecture. Web services interoperate based on a formal definition independent of the underlying platform and programming language. Web services are a collection of technologies driven by standards. Standards are extremely important for the success of Web services. The interoperability of different vendors' Web service implementations rests on the standards process. Generally, web services are a SOA with at least the following additional constraints:

- Interfaces must be based on Internet protocols such as HTTP, FTP, and SMTP (as well as the more recent Blocks Extensible Exchange Protocol (BEEP)).
- Except for binary data attachment, messages must be in XML.

Successful implementation of web services relies on a wide range of specifications, that are used to define, locate, implement, and make web services to interact with each other. The basic web services specifications include the following areas:

- **messaging:** is responsible for encoding messages in a common XML format so that messages can be understood at the either ends of the network connection.
- **service descriptions:** it is used for describing the public interface to a specific web service.
- **service discovery:** centralizes services into a common registry such that network web services can publish their location and description, and makes it easy to discover what services are available on the network.
- **service security:** using the security specifications, applications can engage in secure communication designed to work with the general Web services framework.
- **business process:** specifies the potential execution order of operations from a collection of web services, the data shared between these web services, which partners are involved and how they are involved in the business process, joint exception handling for collections of web services, and other issues involving how multiple services and organizations participate.
- **management:** defines a set of capabilities for discovering the existence, availability, health, performance, usage, as well as the control and configuration of a web service within the web services architecture.
- **profiles:** profiles are designed to give implementers clear guidance on how to address the most common design issues that can arise in order to improve interoperability.

6.3.1 Messaging specifications

6.3.1.1 XML Protocol

XML Protocol (XMLP) [10] from W3C provides simple protocols that can be ubiquitously deployed and easily programmed through scripting languages, XML tools, interactive Web development tools, etc. The goal is a layered system which will directly meet the needs of applications with simple interfaces (e.g. validateCreditCard), and which can be incrementally extended to provide the security, scalability, and robustness required for more complex application interfaces. The XML Protocol Working Group is chartered to design the following four components:

1. An envelope for encapsulating XML data to be transferred in an interoperable manner that allows for distributed extensibility.
2. A convention for the content of the envelope when used for RPC (Remote Procedure Call) applications. The protocol aspects of this should be coordinated closely with the IETF and make an effort to leverage any work they are doing, see below for details.
3. A mechanism for serializing data representing non-syntactic data models such as object graphs and directed labelled graphs based on the data types of XML Schema.
4. A mechanism for using HTTP transport in the context of an XML Protocol. This does not mean that HTTP is the only transport mechanism that can be used for the technologies developed, nor that support for HTTP transport is mandatory. This component merely addresses the fact that HTTP transport is expected to be widely used, and so should be addressed by this Working Group. There will be coordination with the Internet Engineering Task Force (IETF), see Blocks Extensible Exchange Protocol (BEEP) [12].

Furthermore, the following two general requirements must be met by the work produced by this Working Group:

- The envelope and the serialization mechanisms developed by the Working Group may not preclude any programming model nor assume any particular mode of communication between peers.
- Focus must be put on simplicity and modularity and must support the kind of extensibility actually seen on the Web. In particular, it must support distributed extensibility where the communicating parties do not have a priori knowledge of each other.

XMLP was used as a model for analyzing and evaluating protocols for Web services and resulted in the endorsement of SOAP.

6.3.1.2 Simple Object Access Protocol

The Simple Object Access Protocol (SOAP) [6] by W3C organisation [11] is XML-based protocol for exchanging information in a decentralized, distributed environment (soap can be viewed as the successor of XML-RPC protocol [7]). SOAP supports different styles of information exchange, including:

- Remote Procedure Call style (RPC), which allows for request-response processing, where an endpoint receives a procedure oriented message and replies with a correlated response message.
- Message-oriented information exchange, which supports organizations and applications that need to exchange business or other types of documents where a message is sent but the sender may not expect or wait for an immediate response.

SOAP is independent of protocol, language, platform and operating system and it has support for messages incorporating attachments (using the multipart MIME structure).

A SOAP message consists of a SOAP envelope that contains two data structures, the SOAP header and the SOAP body, and information about the name spaces used to define them:

- The SOAP header is optional. When the header is presented, it provides the information about the request defined in the SOAP body, for example: authentication transactional, security, contextual, user profile information, data encoding, or how a recipient of a SOAP message should process the message.
- The body contains a web service request or reply to a request in XML format. These messages are usually defined using the WSDL specification.

SOAP is typically defined over "firewall friendly" protocols such as HTTP and SMTP. SOAP can be used to exchange complete documents or to call a remote procedure.

6.3.1.3 Asynchronous Application Service Protocol for SOAP

The purpose of the OASIS [8] Asynchronous application service protocol ASAP [9] is to create a very simple extension of SOAP that enables generic asynchronous web services or long-running web services. It is intended to integrate asynchronous services across the Internet and provide for their interaction. The integration and interactions consist of control and monitoring of the service. The protocol is intended to be lightweight and easy to implement, so that a variety of devices and situations can be covered.

6.3.1.4 Representational State Transfer

Representational State Transfer (REST) [14] is an architecture style for distributed networked systems such as www. REST is an architectural style, not standard. REST is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use.

A REST web service can be viewed as a SOA based on the concept of resource. Resources are the sources of specific information, each of which can be referred to using a global identifier (an URI). In order to manipulate these resources, components of the network (clients and servers) communicate via a standardized interface (e.g. HTTP) and exchange representations of these resources (the actual documents conveying the information). Any number of connectors (e.g., clients, servers, caches, tunnels, etc.) can mediate the request, but each does so without "seeing past" its own request. Thus an application can interact with a resource by knowing two things: the identifier of the resource, and the action required – it does not need to know whether there are caches, proxies, gateways, firewalls, tunnels, or anything else between it and the server actually holding the information. The application does, however, need to understand the format of the information (representation) returned, which is typically an HTML or XML document of some kind, although it may be an image or any other content [15].

REST has the following characteristics:

- Client-Server: a pull-based interaction style: consuming components pull representations.
- Stateless: each request from client to server must contain all the information necessary to understand the request, and cannot take advantage of any stored context on the server.
- Cache: to improve network efficiency responses must be capable of being labelled as cacheable or non-cacheable.
- Uniform interface: all resources are accessed with a generic interface (e.g., HTTP GET, POST, PUT, DELETE).
- Named resources - the system is comprised of resources which are named using a URL.
- Interconnected resource representations - the representations of the resources are interconnected using URLs, thereby enabling a client to progress from one state to another.
- Layered components - intermediaries, such as proxy servers, cache servers, gateways, etc, can be inserted between clients and resources to support performance, security, etc.

REST design can be described in the terms of verbs, nouns and content types:

- Verbs: The set of well-defined operations. HTTP defines a small set of operations (methods), the most important of which are POST, GET, PUT, DELETE.
- Nouns: Resource identifiers specified as URLs.
- Content types: In HTTP, this is a set of mime types. Each resource may be able to return its state in one of several representations, and accept state updates in the form of one of several representations.

This concept leads to the definition of fewer types (verbs and content types) on the network than an RPC-based application but more resource identifiers (nouns).

An RPC application is exposed as one or more network objects, each with an often unique set of functions that can be invoked. Before a client communicates with the application it must have knowledge of the object identity in order to locate it and must also have knowledge of the object type in order to communicate with it.

REST design seeks to define a set of resources that clients can interact with uniformly, and to provide hyperlinks between resources that clients can navigate without requiring knowledge of the whole resource set. Server-provided forms can also be used environment to describe how clients should construct a URL in order to navigate to a particular resource.

6.3.1.5 Web Distributed Data Exchange

Web Distributed Data eXchange (WDDX) [13] from OpenWDDX.org is a technology for exchanging complex data structures between programming languages. It has been designed with web applications in mind. WDDX consists of a standard for language-independent representation of instantiated data based on XML 1.0 and a set of serializer/deserializer modules for every language/technology that uses WDDX.

6.3.1.6 Web Services Addressing

Web Services Addressing (WS-Addressing) [16] provides transport-neutral mechanisms to address web services and messages. Specifically, this specification defines XML elements to identify web service endpoints and to secure end-to-end endpoint identification in messages. This specification enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner.

6.3.1.7 Web Services Eventing

Web Services Eventing (WS-Eventing) [17] provides a protocol that allows web services to subscribe to or accept subscriptions for event notification messages. This specification defines a protocol for one web service (called an "event sink") to register interest (called a "subscription") with another web service (called an "event source") in receiving messages about events (called "notifications"). To improve robustness, the subscription is leased by an event source to an event sink, and the subscription expires over time. An event source may allow an event sink to renew the subscription.

6.3.1.8 Web Services Notification

The Web Services Notification (WSN) [18] defines a set of specifications that standardize the way web services interact using the notification pattern. In the notification pattern, a web service disseminates information to a set of other web services, without having to have prior knowledge of these other web services. Characteristics of this pattern include:

- The web services that wish to consume information are registered with the web service that is capable of distributing it. As part of this registration process they may provide some indication of the nature of the information that they wish to receive.

- The distributing web service disseminates information by sending one-way messages to the web services that are registered to receive it. It is possible that more than one web service is registered to consume the same information. In such cases, each web service that is registered receives a separate copy of the information.
- The distributing web service may send any number of messages to each registered web service. It is not limited to sending just a single message.

6.3.1.9 Web Services Reliability

The Web Services Reliability (WS-Reliability) is a generic and open model for ensuring reliable message delivery for web services [19]. The reliability features are based on extensions to the Simple Object Access Protocol (SOAP), rather than being tied to the underlying transport protocol. The specification will allow a variety of systems to interoperate reliably in a platform- and vendor-neutral manner.

It defines reliable message delivery as the ability to guarantee message delivery to software applications - Web services or Web service client applications - with a chosen level of quality of service (QoS). QoS is defined as the ability to determine the following aspects of message delivery:

- Message persistence
- Message acknowledgement and resending
- Elimination of duplicate messages
- Ordered delivery of messages
- Delivery status awareness for sender and receiver applications

The WS-Reliability specification provides WSDL definitions for reliable messaging and the message formats specified as SOAP headers or body content.

6.3.1.10 Web Services Reliable Messaging

Web Services Reliable Messaging (WS-ReliableMessaging) [20] describes a protocol that allows messages to be delivered reliably between distributed applications in the presence of software component, system, or network failures. The protocol is described in this specification in an independent manner allowing it to be implemented using different network transport technologies. To support interoperable Web services, a SOAP binding is defined within this specification.

6.3.1.11 Transaction specifications

Web Services Coordination

Web Services Coordination (WS-Coordination) [21] describes an extensible framework for providing protocols that coordinate the actions of distributed applications. Such coordination protocols are used to support a number of applications, including those that need to reach consistent agreement on the outcome of distributed activities.

The framework defined in this specification enables an application service to create a context needed to propagate an activity to other services and to register for coordination protocols. The framework enables existing transaction processing, workflow, and other systems for coordination to hide their proprietary protocols and to operate in a heterogeneous environment.

Additionally this specification describes a definition of the structure of context and the requirements for propagating context between cooperating services.

Web Services Atomic Transaction

Web Services Atomic Transaction (WS-AtomicTransaction) [21] provides the definition of the atomic transaction coordination type that is to be used with the extensible coordination framework described in the WS-Coordination specification. The specification defines three specific agreement coordination protocols for the atomic transaction coordination type: completion, volatile two-phase

commit, and durable two-phase commit. Developers can use any or all of these protocols when building applications that require consistent agreement on the outcome of short-lived distributed activities that have all-or-nothing semantics.

Web Services Business Activity

Web Services BusinessActivity (WS-BusinessActivity) provides the definition of the business activity coordination type that is to be used with the extensible coordination framework described in the WS-Coordination specification. The specification defines two specific agreement coordination protocols for the business activity coordination type: BusinessAgreementWithParticipantCompletion, and BusinessAgreementWithCoordinatorCompletion. Developers can use any or all of these protocols when building applications that require consistent agreement on the outcome of long-running distributed activities.

6.3.2 Service description

6.3.2.1 Web Services Description Language

The Web Services Description Language (WSDL) [22] is an XML language for describing the syntax of web service interfaces and their locations. In accordance to the specification, WSDL is a XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

Programmers or automated development tools can create WSDL files to describe a service and can make the description available over the Internet. Client-side programmers and development tools can use published WSDL descriptions to obtain information about available web services and to build and create proxies or program templates that access available services.

WSDL document has the following elements:

- definitions: Defines one or more services. A definition element supports the following attributes:
 - name is optional.
 - targetNamespace is the logical namespace for information about this service. WSDL documents can import other WSDL documents, and setting targetNamespace to a unique value ensures that the namespaces do not clash.
 - xmlns is the default namespace of the WSDL document, and it is set to `http://schemas.xmlsoap.org/wsdl/`. All the WSDL elements, such as `<definitions>`, `<types>` and `<message>` reside in this namespace.
 - xmlns:xsd and xmlns:soap are standard namespace definitions that are used for specifying SOAP-specific information as well as data types.
 - xmlns:tns stands for this namespace.
- types: Provides information about any complex data types used in the WSDL document. When simple types are used, the WSDL document does not need this section.
- message: An abstract definition of the data being communicated.
- operation: An abstract description of the action supported by the service
- portType: An abstract set of operations supported by one or more endpoints.
- binding: Describes how the operation is invoked by specifying concrete protocol and data format specifications for the operations and messages. The binding for a message defines the wire form of a message, typically in the context of a specific message format standard such as SOAP or MIME.
- port: Specifies a single endpoint as an address for the binding, thus defining a single communication endpoint.

- **service:** Specifies the port address(es) of the binding. The service is a collection of network endpoints or ports.

WSDL is often used in combination with SOAP and XML Schema to provide web services over the internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

An extension of the WSDL is the XLANG [23]. A XLANG service description is a WSDL service description with an extension element that describes the behaviour of the service as a part of a business process.

6.3.2.2 Web Services Semantics

The Web Services Semantics (WSDL-S) [27] specification is a W3C Member Submission that defines how to add semantic information to WSDL documents. Semantic annotations define the meaning of the inputs, outputs, preconditions and effects of the operations described in a service interface. These annotations reference concepts in an ontology. Semantic annotations are used to automate service discovery, composition, mediation, and monitoring.

The semantic information includes definitions of the precondition, input, output and effects of web service operations. This approach offers multiple advantages over OWL-S [28]. First, users can describe, in an upwardly compatible way, both the semantics and operation level details in WSDL - a language that the developer community is familiar with. Second, by externalizing the semantic domain models, an agnostic approach to ontology representation languages is taken. This allows web service developers to annotate their web services with their choice of ontology language (such as UML or OWL) unlike in OWL-S.

6.3.2.3 Web Services Policy Framework

Web Services Policy Framework (WS-Policy) [24] provides a general purpose model and corresponding syntax to describe and communicate the policies of a web service. WS-Policy defines a base set of constructs that can be used and extended by other web services specifications to describe a broad range of service requirements, preferences, and capabilities.

- **WS-PolicyAssertions** express the capabilities and constraints of a particular Web service.
- **WS-PolicyAttachment** specifies three specific attachment mechanisms for using policy expressions with existing XML web service technologies. Specifically, they define how to associate policy expressions with WSDL type definitions and UDDI entities. They also define how to associate implementation-specific policy with all or part of a WSDL portType when exposed from a specific implementation.

The specifications have been updated following the republication of WS-Security Policy, to reflect the constraints and capabilities of web services that are using WS-Security, WS-Trust and WS-SecureConversation. WS-ReliableMessaging Policy has also been republished to express the capabilities and constraints of web services implementing WS-ReliableMessaging.

The updated specifications include the definition of nested assertions which allows for additional granularity when expressing certain domain requirements, i.e., the expression of different algorithm suites for a particular transport binding. These specifications help web services providers and consumers discover the capabilities and constraints that they share to enable interoperability of these services.

6.3.2.4 Web Services Dynamic Discovery

Web Services Dynamic Discovery (WS-Discovery) [25] defines a multicast discovery protocol to locate services. By default, probes are sent to a multicast group, and target services that match return a response directly to the requestor. To scale to a large number of endpoints, the protocol

defines the multicast suppression behaviour if a discovery proxy is available on the network. To minimize the need for polling, target services that wish to be discovered send an announcement when they join and leave the network.

6.3.2.5 Web Services Metadata Exchange

Web Services Metadata Exchange (WS-MetadataExchange) [26] defines three request-response message pairs to retrieve three types of metadata: one retrieves the WS-Policy associated with the receiving endpoint or with a given target namespace, another retrieves either the WSDL associated with the receiving endpoint or with a given target namespace, and a third retrieves the XML Schema with a given target namespace. Together these messages allow incremental retrieval of a web service's metadata.

6.3.2.6 Web Services Endpoint Language

Web Service Endpoint Language (WSEL) is an XML format for the description of non-operational characteristics of service endpoints, like quality-of-service, cost, or security properties.

6.3.3 Discovery specifications

6.3.3.1 Universal Description, Discovery and Integration

The Universal Description, Discovery, and Integration (UDDI) [29] specification defines a 4-tier hierarchical XML schema that provides a model for publishing, validating, and invoking information about web services. XML was chosen because it offers a platform-neutral view of data and allows hierarchical relationships to be described in a natural way. UDDI uses standards-based technologies, such as common Internet protocols (TCP/IP and HTTP), XML, and SOAP. UDDI is a standard web service description format and web service discovery protocol. UDDI registry can contain metadata for any type of service, with best practices already defined for those described by WSDL. There are two types of UDDI registries:

- public UDDI registries that serve as aggregation points for a variety of businesses to publish their services
- private UDDI registries that serve a similar role within organizations.

A UDDI registry consists of the following data structure types:

- **businessEntity** - The top-level XML element in a business UDDI entry, it captures the data partners require to find information about a business service, including its name, industry or product category, geographic location, and optional categorization and contact information. It includes support for "yellow pages" taxonomies to search for businesses by industry, product, or geography.
- **businessService** - The logical child of a businessEntity data structure as well as the logical parent of a bindingTemplate structure, it contains descriptive business service information about a group of related technical services including the group name, a brief description, technical service description information, and category information. By organizing web services into groups associated with categories or business processes, UDDI allows more efficient search and discovery of web services.
- **bindingTemplate** - The logical child of a businessService data structure, it contains data that is relevant for applications that need to invoke or bind to a specific web service. This information includes the web service URL and other information describing hosted services, routing and load balancing facilities, and references to interface specifications.
- **tModel** - Descriptions of specifications for web services or taxonomies that form the basis for technical fingerprints; its role is to represent the technical specification of the web service, making it easier for web service consumers to find web services that are compatible with a particular technical specification. That is, based on the descriptions of the specifications for

web services in the tModel structure, web service consumers can easily identify other compatible web services.

The UDDI Business Registry system consists of three directories to perform three types of searches:

- UDDI white pages: basic information such as a company name, address, and phone numbers, as well as other standard business identifiers.
- UDDI yellow pages: detailed business data, organized by relevant business classifications, such as the NAICS, ISO3166, and UNSPSC classification systems.
- UDDI green pages: containing technical information about Web Services that are exposed by a business, including references to specifications of interfaces for Web Services, as well as support for pointers to various file and URL-based discovery mechanisms.

6.3.3.2 ebXML Registry

The ebXML Registry [30] is similar to UDDI in that it allows businesses to find one another, to define trading-partner agreements, and to exchange XML messages in support of business operations. The goal is to allow all these activities to be performed automatically, without human intervention, over the Internet. The ebXML architecture has many similarities to SOAP/WSDL/UDDI, and some convergence is taking place with the adoption of SOAP in the ebXML transport specification. The ebXML messaging specification is based on SOAP with attachments, but does not use WSDL. ebXML does add security, guaranteed messaging, and compliance with business process interaction specifications.

The ebXML initiative is sponsored by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) [31] and OASIS [8] to research, develop, and promote global standards for the use of XML to facilitate the exchange of electronic business data. A major goal for ebXML is to produce standards that serve the same or similar purpose as EDI, including support for emerging industry-specific XML vocabularies.

6.3.3.3 Web Services Inspection Language

Web Services Inspection Language (WSIL) [32] is a service discovery mechanism that is an alternative to UDDI as well as complementary to UDDI. Web services discovery with UDDI is realised using a centralized registry. WSIL is an alternative approach to web service discovery. WSIL allows to access the service provider directly and to ask for the services it provides. IBM's and Microsoft's proposal for the WSIL specification is designed around an XML-based model for building an aggregation of references to existing Web service descriptions, that are exposed using standard web server technology.

Since WSIL focuses on distributed service discovery, the WSIL specification complements UDDI by facilitating the discovery of services that are available on web sites that may not be listed yet in a UDDI registry.

The WSIL specification serves two primary functions:

- WSIL defines an XML format for listing references to existing service descriptions. These service descriptions can be defined in any format, such as WSDL, UDDI, or plain HTML. A WSIL document is generally made available at the point-of-offering for the services that are referenced within the document. A WSIL document can contain a list of references to service descriptions, as well as references to other WSIL documents. The ability to link a WSIL document to one or more different WSIL documents allows you to manage service description references by grouping them into different documents and to build a hierarchy of WSIL documents. For example, separate WSIL documents can be created for different categories of services, and one primary WSIL document can link all of them together.
- WSIL defines a set of conventions so that it is easy to locate other WSIL documents. The WSIL specification does not limit the type of service descriptions that can be referenced. The

WSIL specification defines a set of standard extensibility elements for both WSDL and UDDI. The WSIL specification is the definition of locations where you can access WSIL documents.

6.3.4 Security specifications

6.3.4.1 Extensible Access Control Markup Language

Extensible Access Control Markup Language (XACML) [35] provides fine grained control of authorized activities, the effect of characteristics of the access requestor, the protocol over which the request is made, authorization based on classes of activities, and content introspection.

6.3.4.2 Extensible Rights Markup Language

Extensible rights Markup Language (XrML) is a digital rights language designed for securely specifying and managing rights and conditions associated with various resources including digital content as well as services.

6.3.4.3 Security Assertion Markup Language

Security Assertion Markup Language (SAML) [34] is an XML framework for exchanging authentication and authorization information.

6.3.4.4 Username Token Profile

Web service consumer can supply a UsernameToken [38] as a means of identifying the requestor by "username", and optionally using a password (or shared secret, or password equivalent) to authenticate that identity to the web service producer.

6.3.4.5 X.509 Certificate Token Profile

An X.509 [37] certificate specifies a binding between a public key and a set of attributes that includes (at least) a subject name, issuer name, serial number and validity interval. This binding may be subject to subsequent revocation advertised by mechanisms that include issuance of CRLs, OCSP tokens or mechanisms that are outside the X.509 framework, such as XKMS. An X.509 certificate may be used to validate a public key that may be used to authenticate a SOAP message or to identify the public key with SOAP message that has been encrypted.

6.3.4.6 Security Kerberos Binding

Kerberos Binding (WS-SecurityKerberos) [39] describes how to use web services security specifications with Kerberos. Kerberos is an established authentication and security infrastructure in use in many environments today. Consequently, as applications integrate with and are developed for web services, there is a need to leverage existing security infrastructure.

Integration with web services security requires the following aspects:

- Requesting and issuing security tokens
- Attaching security token to messages
- Establishing a secure context
- Signing and encrypting the message using the security context

6.3.4.7 Web Services Security

Web Services Security (WSS or WS-Security) [33] describes enhancements to SOAP messaging in order to provide quality of protection through message integrity, and single message authentication.

These mechanisms can be used to accommodate a wide variety of security models and encryption technologies.

The scope of the Web Services Security Technical Committee is the support of security mechanisms in the following areas:

- Using XML Signature to provide SOAP message integrity for web services
- Using XML Encryption to provide SOAP message confidentiality for web services
- Attaching or referencing security tokens in headers of SOAP messages. Options include:
 - Username token
 - SAML
 - XrML
 - Kerberos
 - X.509
- Carrying security information for potentially multiple, designated actors.
- Associating signatures with security tokens.
- Each of the security mechanisms will use implementation and language neutral XML formats defined in XML Schema.

6.3.4.8 Web Services Provisioning

Web Services Provisioning (WS-Provisioning) [36] describes the APIs and schemas necessary to facilitate interoperability between provisioning systems and to allow software vendors to provide provisioning facilities in a consistent way. The specification addresses many of the problems faced by provisioning vendors in their use of existing protocols, commonly based on directory concepts, and confronts the challenges involved in provisioning Web Services described using WSDL and XML Schema.

6.3.4.9 Web Services Security Policy

Web Services Security Policy Language (WS-SecurityPolicy) [41] specification defines a set of security policy assertions which apply to web services security: SOAP Message Security, WS-Trust, and WS-SecureConversation. The specification takes the approach of defining a base set of assertions that describe how messages are to be secured. Flexibility with respect to token types, cryptographic algorithms, and mechanisms used, including using transport-level security, is part of the design and allows for evolution over time. The intent is to provide enough information for compatibility and interoperability to be determined by web services participants, along with all information necessary to actually enable a participant to engage in a secure exchange of messages.

6.3.4.10 Web Services Trust

The Web Services Trust Language (WS-Trust) [40] uses the secure messaging mechanisms of WS-Security to define additional primitives and extensions for the issuance, exchange and validation of security tokens. WS-Trust also enables the issuance and dissemination of credentials within different trust domains.

In order to secure a communication between two parties, the two parties must exchange security credentials (either directly or indirectly). However, each party needs to determine if they can “trust” the asserted credentials of the other party. This specification defines extensions to WS-Security for issuing and exchanging security tokens and ways to establish and access the presence of trust relationships. Using these extensions, applications can engage in secure communication designed to work with the general web services framework, including WSDL service descriptions, UDDI businessServices and bindingTemplates, and SOAP messages.

6.3.4.11 Web Services Secure Conversation

The Web Services Secure Conversation Language (WS-SecureConversation) [42] is built on top of the WS-Security and WS-Policy models to provide secure communication between services. WS-Security focuses on the message authentication model but not a security context, and thus is subject several forms of security attacks. WS-SecureConversation specification defines mechanisms for establishing and sharing security contexts, and deriving keys from security contexts, to enable a secure conversation.

By using the SOAP extensibility model, modular SOAP-based specifications are designed to be composed with each other to provide a rich messaging environment. As such, WS-SecureConversation by itself does not provide a complete security solution. WS-SecureConversation is a building block that is used in conjunction with other web service and application-specific protocols (for example, WS-Security) to accommodate a wide variety of security models and technologies.

6.3.4.12 Web Services Federation

The Web Services Federation Language (WS-Federation) specification is another component of the web services security model that defines mechanisms to allow different security realms to federate by allowing and brokering trust of identities, attributes, authentication between participating web services. The mechanisms defined in this specification can be used by passive and active requestors. The web service requestors are assumed to understand the new security mechanisms and be capable of interacting with web service providers.

- **Active Requestor Profile:** By using the XML, SOAP and WSDL extensibility models, the WS* specifications are designed to be composed with each other to provide a rich web services environment. WS-Federation: Active Requestor [45] by itself does not provide a complete security solution for web services. WS-Federation: Active Requestor is a building block that is used in conjunction with other web service and application-specific protocols to accommodate a wide variety of security models.
- **Passive Requestor Profile:** The WS-Federation specification defines an integrated model for federating identity, authentication and authorization across different trust realms and protocols. WS-Federation: Passive Requestor specification defines how the WS-Federation model is applied to passive requestors such as web browsers that support the HTTP protocol. For the passive mechanisms to work seamlessly with WS-Federation, and provide a single or reduced sign-on, there needs to be a service that will verify that the claimed requestor is really the requestor. Initial verification MUST occur in a secure fashion, for example, using SSL/TLS or HTTP/S.

6.3.4.13 XML Common Biometric Format

XML Common Biometric Format (XCBF) [43] is a common set of secure XML encoding for the formats specified in CBEFF, the Common Biometric Exchange File Format.

6.3.4.14 XML Key Management Specification

XML Key Management Specification (XKMS) [44] is a specification of XML application/protocol that allows a simple client to obtain key information (values, certificates, and management or trust data) from a web service.

6.3.5 Business process specifications

6.3.5.1 Business Process Execution Language

Business Process Execution Language (WS-BPEL) [46] defines a notation for specifying business process behaviour based on Web Services. Business processes can be described in two ways:

- Executable business processes model actual behaviour of a participant in a business interaction.
- Business protocols, in contrast, use process descriptions that specify the mutually visible message exchange behaviour of each of the parties involved in the protocol, without revealing their internal behaviour. The process descriptions for business protocols are called abstract processes.

WS-BPEL is used to model the behaviour of both executable and abstract processes. The scope includes:

- Sequencing of process activities, especially web service interactions
- Correlation of messages and process instances
- Recovery behaviour in case of failures and exceptional conditions
- Bilateral web service based relationships between process roles

6.3.5.2 WS-BPEL Extension for People

Human user interactions are currently not covered by the web services business process Execution Language (WS-BPEL), which is primarily designed to support automated business processes based on web services. In practice, however, many business process scenarios require user interaction. WS-BPEL extension for people (BPEL4People) [50] describes scenarios where users are involved in business processes and then defines appropriate extensions to WS-BPEL which address these scenarios.

BPEL4People is defined in a way that it is layered on top of the BPEL language so that its features can be composed with the BPEL core features whenever needed. Additional BPEL extensions might be introduced that can use the BPEL4People extension introduced here.

6.3.6 Management specifications

6.3.6.1 Web Services Management

Web Services Management (WS-Management) [51] specification describes a general SOAP-based protocol for managing systems such as PCs, servers, devices, web services, other applications, and other manageable entities.

To promote interoperability between management applications and managed resources, WS-Management specification identifies a core set of web service specifications and usage requirements to expose a common set of operations that are central to all systems management. This comprises the abilities to:

- DISCOVER the presence of management resources and navigate between them.
- GET, PUT, CREATE, RENAME, and DELETE individual management resources, such as settings and dynamic values.
- ENUMERATE the contents of containers and collections, such as large tables and logs.
- SUBSCRIBE to events emitted by managed resources.
- EXECUTE specific management methods with strongly typed input and output parameters.

In each of these areas of scope, the specification defines minimal implementation requirements for conformant web service implementations. An implementation is free to extend beyond this set of operations, and may also choose not to support one or more areas of functionality listed above if that functionality is not appropriate to the target device or system.

6.3.6.2 Web Services Management Catalog

WS-Management is a general-purpose SOAP-based systems management protocol and is based on a small number of fixed operations typical to management tasks. Web Services Management Catalog (WS-Management Catalog) [52] specification defines the default metadata formats for the discovery part of the protocol. Discovery in this context refers to discovering available resources at a particular network address or management node.

WS-Management supports the concept of multiple logical endpoints residing at the same network address, so a technique is required for discovering and understanding what management functionality those endpoints represent. This list of available logical endpoints or "resources", their summary forms, compatible actions, schemas, and WSDL representations loosely constitute the WS-Management Catalog.

While WS-Management itself can work with more than one metadata format, WS-Management Catalog specification is offered as a practical starting point for organizing the management data needed by users of the protocol.

6.3.6.3 Web Services Distributed Management

Web Services Distributed Management (WSDM) is a standard that seeks to unify management infrastructures by providing a vendor, platform, network, and protocol neutral framework for enabling management technologies to access and receive notifications of management-enabled resources. Though built upon a standardized suite of XML specifications, it provides features to enable resources that other proprietary management technologies do not. It can be used to standardize management for many devices, from network management devices as well as consumer electronic devices, such as televisions, digital video disc players, and PDAs.

6.3.7 Specification profiles

6.3.7.1 Devices Profile for Web Services

Devices Profile for Web Services [47] defines a minimal set of implementation constraints to enable secure web service messaging, discovery, description, and eventing on resource-constrained endpoints.

The web services architecture includes a suite of specifications that define rich functions and that may be composed to meet varied service requirements. To promote both interoperability between resource-constrained web service implementations and interoperability with more flexible client implementations, this profile identifies a core set of web service specifications in the following areas:

- Sending secure messages to and from a web service
- Dynamically discovering a web service
- Describing a web service
- Subscribing to, and receiving events from, a web service

In each of these areas of scope, this profile defines minimal implementation requirements for compliant web service implementations.

6.3.7.2 WS-I Basic Profile

Today's web services are composed from a number of technologies such as SOAP and WSDL. Many of these technologies provide different variations of features or multiple approaches for the same feature. For example, different SOAP implementations use different HTTP status codes for HTTP responses to one-way messages, which may cause interoperability problems.

The WS-I Basic Profile [48] defines an interoperable subset of the core web services specifications, including XML Schema, SOAP, WSDL, and UDDI, by specifying refinements, interpretations, and clarifications of these specifications.

6.3.8 Semantic Web and Web Services

The semantic enhancements of web services if strongly linked to the ontologies domain (and so the semantic web). The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF) [53], which integrates a variety of applications using XML for syntax and URIs for naming.

The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation [54].

Generally, the Semantic web is about making explicit the meaning of web resources, allowing smart search, allowing software agents to co-operate. Semantic web technology can be used to support various forms of collaboration and knowledge work for: retrieval, publishing, interpretation, personalisation, knowledge sharing. Semantic web technology will facilitate the emergence of hybrid communities, e.g., software agents contribute to the dialectic of a scientific or scholarly community.

One of the important points about web services is that they consume and produce XML. Thus, the first way that web services fit into the semantic web is by furthering the adoption of XML, or more smart data. As web services proliferate, they become similar to web pages because they are more difficult to discover. Semantic web technologies will be necessary to solve the web service discovery problem. There are several research efforts under way to create semantic web-enabled web services. Figure 4 demonstrates the various convergences that combine to form semantic web services.

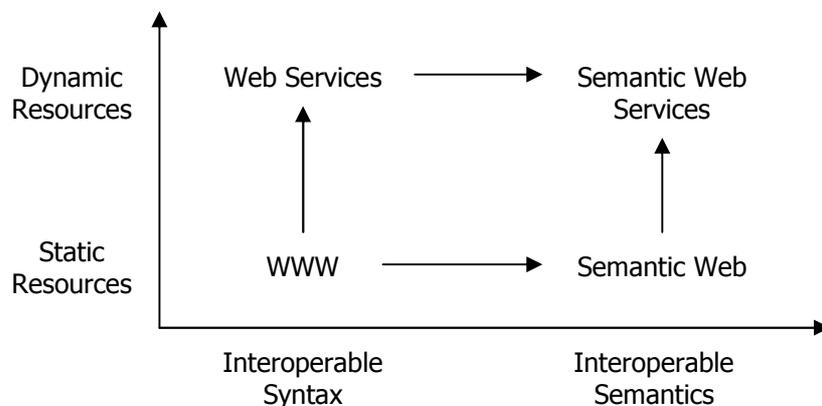


Figure 4 Semantic web services

Another way that web services fit into the semantic web is in enabling web services to interact with other web services. Advanced web service applications involving comparison, composition, or orchestration of web services will require semantic web technologies for such interactions to be automated.

6.4 SOA quality aspects

SOA quality is fast becoming one of the next critical issues that enterprises, consulting firms, and vendors alike must face when implementing SOA [59]. SOA quality is a relatively new concept. In the case of Web services, quality was associated with simply testing a service. Individual Web services have long been treated as traditional software applications, commonly tested with a waterfall approach. This approach works for a Web service because services have a distinct function and their own lifecycle.

On the other hand, SOA is not software. A SOA does not have a lifecycle. A SOA is an architecture, made up of infrastructure and services that must constantly interoperate with each other. It does not go off-line and cannot be replaced. It can, however, evolve and expand. It can also improve or degrade. Therefore, the quality of an SOA is reflected by how well its implementation meets the needs of the business even as those needs evolve.

SOA quality is a key component of reaping the intended benefits of an SOA - it's the strategy needed to achieve maximum business benefit.

6.4.1 Quality requirements

The choice to use an SOA approach in the development of an architecture depends on several factors including the architecture's ultimate ability to meet functional and quality attribute requirements [55]. Usually, an architecture needs to satisfy many quality attribute requirements in order to achieve the organization's business goals.

6.4.1.1 Modifiability

Modifiability is the ability to make changes to a system quickly and cost-effectively. SOA promotes loose coupling between service consumers and providers. Services are self-contained, modular, and accessed via cohesive interfaces. These characteristics contribute to the creation of loosely coupled SOAs where there are few, well-known dependencies between services. That fact tends to reduce the cost of modifying the implementation of services, hence increasing the system's modifiability.

Modifiability can be regarded as the attribute with the closest connection to architecture. This, mainly because the attribute focuses on to what extent certain attributes within the architecture can be modified. In other words, modifiability is not about the change of the overall architecture, but rather the change of processes, products, technologies, behaviour (rules) etc.

6.4.1.2 Portability

The attribute modifiability evaluates to what extent it is possible to modify attributes of the architecture without affecting the overall architectural structure. Portability on the other hand, evaluates if the overall architecture can be moved to another environment, if it is adaptable and replaceable.

One of the strengths of SOA evolves around platform independency. That means, platform specific information is being encapsulated and hidden behind an abstract interface, offering portability in terms of transparency to the system being based upon SOA.

6.4.1.3 Reusability

Reusability is an attribute which questions to what extent different system components can be reused, either within the same or in another system. Reused in the sense that the components do not have to go through any changes, but can simply be used the way they are and have been defined.

There is also an indirect relationship between modifiability and reusability. Changes or modifications, for example, might turn out to be either efficiently or simply inefficiently conducted, all depending on whether or not the components are loosely or strongly coupled.

6.4.1.4 Integrability

Integrability is the quality attribute covering integration between two or more components and services of a system. Together with this concept usually also interoperability is mentioned, to highlight the possibility that not only separate components need to be integrated but also, for example, groups of parts with other old or new systems. The ability of integrating loosely-coupled components or services depends on the external complexity of the components/services, the

interaction mechanisms, the protocols used, as well as all other issues being typical for each architectural level.

6.4.1.5 Security

The security aspect in combination with SOAs become especially important due to the introduction of Web Services. SOA does not have to be implemented only as a set of Web Services, but still this is a very SOA specific feature. Generally, this quality attribute is mainly about providing architectures with prevention of unauthorized access, both accidental and deliberate.

Although security denotes different things with respect to software systems, in general, it is associated with four basic principles:

- confidentiality: access to information/service is granted only to authorized subjects.
- authenticity: trust that the indicated author/sender is the one responsible for the information.
- integrity: information is not corrupted.
- availability: the information/service is available in a timely manner.

6.4.1.6 Performance

In general, performance quality attribute is related to response time (how long it takes to process a request), throughput (how many requests overall can be processed per unit of time), or timeliness (ability to meet deadlines, i.e., to process a request in a deterministic and acceptable amount of time). Performance is an important quality attribute that is usually affected negatively in SOAs. Careful design and evaluation of the architecture for the specific solution is necessary to avoid performance falls. The key factors in SOA that contribute to performance issues are:

- SOA involves distributed computing. Service and service user components are normally located in different containers, most often on different machines. The need to communicate over the network increases the response time.
- The interaction protocol sometimes requires a call to a directory of services to locate the desired service. This extra call increases the total time needed to perform the transaction.
- The ability to make services on different platforms interoperate seamlessly has a performance cost. Intermediaries are needed to perform data marshalling and handle all communication between a service user and a service provider.
- The use of a standard messaging format increases the time needed to process a request.

Many SOA technologies permit the service user to call the provider asynchronously. In that case, the user does not get blocked waiting for the response. For operations that fit that model of interaction, asynchronous calls should be used to reduce the response time.

The other architectural aspect, affecting the efficiency, is the way the business is structured, i.e. how efficiently the business processes are. As long as these processes are not optimized, the system will be seen as inefficient.

6.4.1.7 Scalability

Scalability is the ability of an SOA to function well (without degradation of other quality attributes) when the system is changed in size or in volume in order to meet users' needs [56]. One of the major issues in scalability is the capacity of the site where the services are located to accommodate an increasing number of service users without a degradation of the services' performance.

Because service users know only about the service's interface and not its implementation, changing the implementation to be more scalable requires little overhead. In general, options for solving scalability problems include:

- horizontal scalability: distributing the workload across more computers. Doing so may mean adding an additional tier or more service sites.
- vertical scalability: upgrading to more powerful hardware for the service site

6.4.1.8 Reliability

Reliability is the ability of a system to keep operating over time. Several aspects of reliability are important within an SOA, particularly the reliability of the messages that are exchanged between the application and the services, and the reliability of the services themselves. Applications developed by different organizations may have different reliability requirements for the same set of services. And an application that operates in different environments may have different reliability requirements in each one. The reliability aspect covers two key elements:

- message reliability: Services are often made available over a network with possibly unreliable communication channels. Connections break and messages fail to get delivered or are delivered more than once or in the wrong sequence.
- service reliability: Service reliability means the service operates correctly and either does not fail or reports any failure to the service user. Service reliability also means making sure that the service is obtained from a reliable provider so that a level of trust in the service's accuracy and reliability can be established.

6.4.1.9 Availability

Availability is the degree to which a system or component is operational and accessible when required for use. Availability of services both from the user's and provider's perspectives is a concern for the success of an SOA. From the services user's perspective, if the system relies on a set of services being available in order to meet its functional requirements and one of those services becomes unavailable (even transiently), it could have dire consequences on the success of the system. From the service provider's perspective, in order for the services to be used (for which the provider may receive compensation), they must be available when needed. Otherwise, the provider's finances and reputation could be impacted (especially if compensation has to be paid when the services are not available).

6.4.1.10 Testability

Testability is the degree to which a system or service facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met. Testing a system that uses an SOA can be complex for many reasons including:

- Interactions may be required between distributed pieces of the system (i.e., pieces that run on different machines across a network).
- The organization may not be able to access the services' source code, so it can't identify the test cases required to thoroughly test them. This problem occurs when the services are external to the organization that owns the applications.
- Services may be discovered at runtime, so it may be impossible to predict which service or set of services is actually used by a system until the system is executing. In addition, different services from different providers may be used at various times when the system runs. The services used may be running on different platforms or operating systems and use different middleware technologies. Building repeatable tests and automating the testing process for such a system will be a challenge.

There are many potential sources for the problem, and trying to replicate it in a test environment may be extremely challenging, if not impossible, because the service is provided by an outside source—a fact the service user can't change. Service providers may need to build additional services and infrastructure that support the testing and debugging processes of both the service and the service users.

6.4.2 The foundations of SOA quality

SOA quality requires a solid foundation. Achieving business objectives from an SOA requires both top-down approach to mapping out those objectives and simultaneously a bottom-up strategy for SOA quality based on a solid foundation [57].

There are five major components that make up the foundation of SOA quality, and all must be present for a solid foundation. As with any foundation, if one of the components is weak or is missing, the strategy is at risk.

6.4.2.1 Prototyping

A key component to quality in an SOA is the ability to prototype. Prototyping is one of the best ways to reach agreement on a WSDL contract before any code is written. Prototyping lets business analysts, architects and developers design and develop very usable interfaces early in the process, allowing them to create services designed for reuse. It also allows consumers and testers to get involved much earlier in the design process, reducing the overall development cycle. It accelerates development time and reduces the testing cycles and resource requirements. Prototyping contributes to reuse by ensuring that the right services are available to meet business needs which helps them get reused.

6.4.2.2 Compliance

The SOA quality should start with compliance to standards. Non-compliant services pose the high risk for SOA quality, and simply cannot exist if an SOA is to be successful. Even well-written services cannot guarantee broad interoperability unless standards and best practices are well designed and adhered to throughout an organization.

One example of how compliance and quality are being applied early in the process is a "Contract-first" approach [58]. Instead of developing Web services first and then letting the development environment generate the interfaces for the services, contract-first design involves reviewing the required interfaces first, which then drives the development process. Contract-first SOA design can reduce development cycles and improve interoperability and compliance early and support ongoing compliance over time.

6.4.2.3 Testing

Traditional software testing that focused on code-level testing has evolved with web applications. Web application testing has introduced more testing of business logic through the application's user interface, which has proved to be critical when deploying new solutions. With SOA, the need to test the business logic still exists, and is even more important with disparate services. However, Web services lack a user interface, making it very challenging to test the business logic within an SOA and ensure that these applications can support the business.

SOA has many moving parts. It is practically impossible to test every service and its interaction with its dependencies within an SOA. Testing also occurs while many services are in various stages of development, including production. New testing methods are needed in order to fulfill the unique aspects of an SOA.

6.4.2.4 Diagnostics

Determining if a service can perform its intended function is often a time-sensitive issue that might require fast identification of a root cause problem. This can pose significant challenges to many teams and individuals because services in an SOA are complex. With an SOA, problems often need to be solved in real time, and might involve disparate teams and systems, what requires some kind of collaborative diagnostics.

6.4.2.5 Support

In the SOA, reproducing problems is a major challenge. Support will often involve many groups to solve a problem. Support is fundamentally different in an SOA because it involves two phases that span design time and run time simultaneously. As services are exposed for use and reuse, consumers will require support to assist in the development of their applications. Automated "pre-support" involves exposing documented contracts and providing a way for team members to investigate services and try different scenarios. In production, support teams need to understand and resolve problems quickly and often need to reproduce scenarios when a failure occurred. When service developers need to get involved, complete problem data needs to be shared with other team members with the ability to simulate different scenarios, to more effectively diagnose problems.

6.4.3 The key elements of SOA quality

6.4.3.1 Communication

The ability to communicate effectively is an essential core element in a successful SOA. It is equally important to SOA technologies as well as to the people involved with the environment. Communication in a technology domain is often referred to as "interoperation," and occurs across different platforms, languages, locations, policies, and standards. There also needs to be communication between people, because individual team members depend upon each other in order to do their job, complete a task, solve a problem, or contribute to a project.

Communication is more than just interoperability and collaboration. It involves the interaction between people and technologies, which is often where quality breaks down in an SOA. Visibility is an important attribute offered by a registry, providing a catalog of services that are available for use. Accessibility and understanding together help make visibility a form of communication in an SOA.

6.4.3.1.1 Trust

The primary characteristics of SOA quality for any business are agility and reuse. Trust is what drives service reuse. It is critical for SOA teams to be aware of existing services in the SOA, understand what they can and cannot do, and most importantly, have trust and confidence that the services will execute as intended. Trust is relevant in nearly every aspect of an SOA. Architects need to trust the services they choose to use. Developers need to trust that an existing service will be appropriate for an application, versus building a new one. They must also trust the policies and standards that have been implemented are meaningful and add value. SOA leaders must trust that services are being added to the registry and reused.

Without trust, an SOA will never achieve reuse, which leads to redundant services or rogue services. As services gain trust, users will share their positive experience with peers-the trust factor grows and reuse increases. SOA quality optimization depends upon creating trusted services.

6.4.3.2 Control

The control aspect of governance solutions is a critical element of SOA quality. Control is essential at design time, change time and run time. It involves policy enforcement to ensure quality, and requires changes in human behaviour, development concepts, and process. Control also involves reducing the number of production issues during run time and resolving those issues quickly. Further, control involves understanding the dependencies applications have on multiple, disparate services, and prompts the need to test services that depend on other services that might not be directly accessible.

6.5 References

- [1] http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html
- [2] <http://www.w3.org/TR/ws-gloss/>

-
- [3] <http://www.w3.org/TR/soap/>
 - [4] <http://webservices.xml.com/pub/a/ws/2002/02/20/rest.html>
 - [5] <http://www.serviceorientation.org/>
 - [6] <http://www.w3.org/TR/soap12-part0/>
 - [7] <http://www.xmlrpc.com/>
 - [8] <http://www.oasis-open.org/>
 - [9] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=asap
 - [10] <http://www.w3.org/2004/02/XML-Protocol-Charter>
 - [11] <http://www.w3.org/>
 - [12] <http://www.beepcore.org/>
 - [13] <http://www.openwddx.org/>
 - [14] http://www1.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
 - [15] <http://rest.blueoxen.net/cgi-bin/wiki.pl?RestInPlainEnglish>
 - [16] <http://www.w3.org/Submission/ws-addressing/>
 - [17] <http://www.w3.org/Submission/WS-Eventing/>
 - [18] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn
 - [19] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsm
 - [20] <http://www-128.ibm.com/developerworks/library/specification/ws-rm/>
 - [21] <http://www-128.ibm.com/developerworks/library/specification/ws-tx/>
 - [22] <http://www.w3.org/TR/wsd/>
 - [23] http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm
 - [24] <http://www.w3.org/TR/ws-policy/>
 - [25] <http://schemas.xmlsoap.org/ws/2005/04/discovery/>
 - [26] <http://schemas.xmlsoap.org/ws/2004/09/mex/>
 - [27] <http://www.w3.org/Submission/WSDL-S/>
 - [28] <http://www.daml.org/services/>
 - [29] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec
 - [30] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=regrep
 - [31] <http://www.unece.org/cefact/>
 - [32] <http://www-128.ibm.com/developerworks/library/specification/ws-wsilspec/>
 - [33] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
 - [34] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
 - [35] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
 - [36] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=provision
 - [37] <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>
 - [38] <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf>
 - [39] <http://msdn.microsoft.com/webservices/webservices/understanding/specs/default.aspx?pull=/library/en-us/dnglobspec/html/ws-security-kerberos.asp>
 - [40] <http://msdn.microsoft.com/ws/2005/02/ws-trust/>

-
- [41] <http://msdn.microsoft.com/ws/2005/07/ws-security-policy/>
 - [42] <http://msdn.microsoft.com/ws/2005/02/ws-secure-conversation/>
 - [43] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xcbf
 - [44] <http://www.w3.org/2001/XKMS/>
 - [45] <http://msdn.microsoft.com/ws/2003/07/ws-active-profile/>
 - [46] <http://msdn.microsoft.com/ws/2003/07/ws-passive-profile/>
 - [47] <http://specs.xmlsoap.org/ws/2006/02/devprof/DevicesProfile.pdf>
 - [48] <http://www.ws-i.org/Profiles/BasicProfile-1.0.html>
 - [49] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
 - [50] <http://www-128.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/>
 - [51] <http://msdn.microsoft.com/ws/2005/08/ws-management/>
 - [52] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm
 - [53] <http://www.w3.org/RDF/>
 - [54] T. Berners-Lee, J. Hendler, O. Lassila, "The Semantic Web", Scientific American, May 2001.
 - [55] Liam O'Brien, Len Bass, Paulo Merson, Quality Attributes and Service-Oriented Architectures, Software Architecture Technology Initiative, Technical note CMU/SEI-2005-TN-014, 2005
 - [56] Worldwide Web Consortium (W3C). Web Services Glossary. <http://www.w3.org/TR/ws-gloss/>, 2004
 - [57] The Foundation of SOA Quality, Mindreef White Paper, Mindreef Inc., 2006 http://www.webservices.org/recommended/the_foundation_of_soa_quality
 - [58] Ronald Schmelzer, Understanding the Relationships among Services, Contracts, and Policies, <http://www.zapthink.com/report.html?id=ZAPFLASH-200628>, 2006
 - [59] Jason Bloomberg, SOA Consulting: Current Market Trends, ZapThink foundation report, 2006

7 Model-driven Architecture

The Model-Driven Architecture (MDA), developed by Object Management Group (OMG) [1], defines an approach to IT systems specifications which separates the business and application logic from the implementation details on a particular technological platform, so that:

- changes in the underlying platform do not affect existing applications, and
- business logic can evolve independently from the underlying technology [2, 3]

The MDA framework allows developers to produce models of the application and business logic and generate code for a target platform by means of transformations. The main benefit of this approach is that it raises the level of abstraction in software development. Developers can focus on design of models that are specific to the application domain, but independent of the target platform, instead of writing platform-specific code.

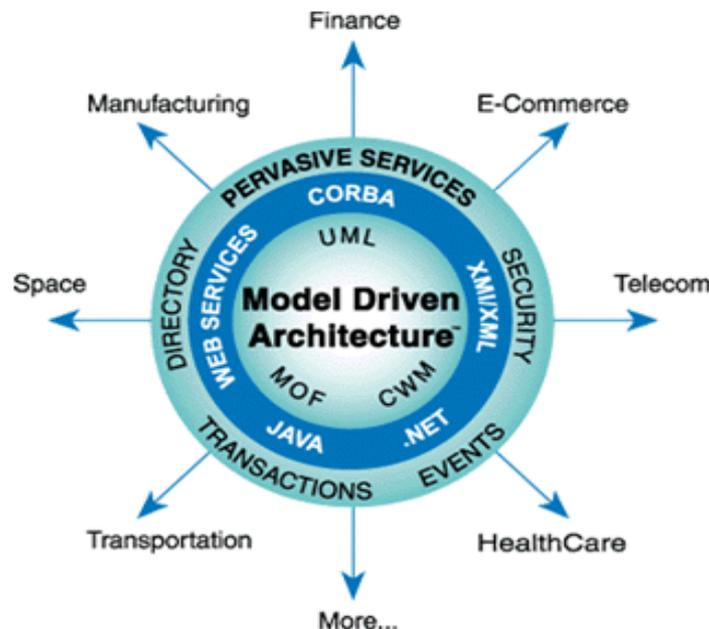


Figure 5 The Model-Driven Architecture (from [1])

The result of the modelling process is so a called Platform Independent Model (PIM), which contains only business and application logic. The code generation process is realised as a set of transformations that map elements in the PIM to the elements in Platform Specific Model (PSM). PSM contains details specific to the target platform. In this way, instead of hand-written platform specific code (conventional way), the result code is generated as a form of PSM. Mapping from PIM to PSM, using transformations, can be done for any particular platform (once a PIM is built), thus, it is not necessary to repeat the modelling process each time, when some new technology emerges.

Platform independent applications build using the MDA framework can be deployed on a range of open platforms, such as CORBA, J2EE, .NET, etc. (see Figure 5 [1]).

OMG claims the following benefits for MDA [4]:

- Reduced cost throughout the application life-cycle
- Reduced development time for new applications
- Improved application quality
- Increased return on technology investments
- Rapid inclusion of emerging technology benefits into their existing systems

If the application requires deployment on particular platform or needs migration from one platform to another (as a technology changes), the specific code can be regenerated from PIM, what is faster and cheaper than the migration of the deployed code. The MDA concept is a step to the cross platform interoperability, portability and platform independence. MDA framework is the paradigm that shifts the system development from coding to modelling. Time and costs spent on coding process should be used in the modelling phase [5].

The promise of MDA is to facilitate the creation of machine-readable models with a goal of long-term flexibility in terms of:

- Technology obsolescence: new implementation infrastructure can be more easily integrated and supported by existing designs.
- Portability: existing functionality can be more rapidly migrated into new environments and platforms as dictated by the business needs.
- Productivity and time-to-market: by automating many tedious development tasks architects and developers are freed up to focus their attention on the core logic of the system.
- Quality: the formal separation of concerns implied by this approach plus the consistency and reliability of the artefacts produced all contribute to the enhanced quality of the overall system.
- Integration: the production of integration bridges with legacy or external systems is greatly facilitated.
- Maintenance: the availability of the design in a machine-readable form gives analysts, developers and testers direct access to the specification of the system, simplifying their maintenance chores.
- Testing and simulation: models can be directly validated against requirements as well as tested against various infrastructures. They can also be used to simulate the behaviour of the system under design.
- Return on investment: businesses are able to extract greater value out of their investments in tools.

7.1 The MDA standards

According to [1], the key standards used in MDA framework include the Unified Modelling Language (UML) [10], Meta Object Facility (MOF) [11], XML Metadata Interchange (XMI) [14] and Common Warehouse Model (CWM) [15].

7.1.1 Unified Modelling Language

In accordance to [9], the UML is an object modelling and specification language, that includes standardized graphical notation that may be used to create an abstract model of a system, sometimes referred to as the UML model. UML is general-purpose modelling language. UML was designed to specify, visualize, construct and document software systems, but it is not restricted only to modelling the software. UML is often used for business process modelling, representing organizational structures, etc. UML enables to represent common concepts like classes, components, generalization, aggregation, behaviours, etc. using the graphic notation. UML allows developers to concentrate more on the design and architecture. UML can be used for designing of models in PIM.

7.1.2 Meta-Object Facility

In accordance to [13], Meta-Object facility is the foundation OMG's industry-standard environment where models can be exported from one application, imported into another, transported across a network, stored in a repository and then retrieved, rendered into different formats (including XMI), transformed, and used to generate application code. These functions are not restricted to structural models, or even to models defined in UML - behavioural models and data models also participate in

this environment, and non-UML modelling languages can partake also, as long as they are MOF-based. MOF concept was introduced as a meta-modelling architecture to define the UML [12].

The MOF specification was developed to provide an open-ended information modelling capability. The specification defines a core MOF model, which contains a set of constructs for object-oriented information modelling. The model can be extended by inheritance and composition to define the wider model containing the specific additional constructs.

7.1.3 XML Metadata Interchange

XML Metadata Interchange (XMI) is an OMG standard for exchanging MDA artefacts serialized in XML, i.e., various metadata elements encoded in XML. It can be used for any metadata whose meta-model can be expressed in Meta Object Facility. The most common use of XMI is as an interchange format for UML models, although it can also be used for serialization of models of other languages (meta-models) [16]. XMI is a model driven XML Integration framework for defining, interchanging, manipulating and integrating XML data and objects. XMI-based standards are in use for integrating tools, repositories, applications and data warehouses. XMI provides rules by which a schema can be generated for any valid XMI-transmissible MOF-based meta-model. XMI provides a mapping from MOF to XML. As MOF and XML technology evolved, the XMI mapping is being updated to comply with the latest versions of these specifications. Updates to the XMI mapping have tracked these version changes in a manner consistent with the existing XMI Production of XML Schema specification (XMI Version 2).

7.1.4 Common Warehouse Meta-model

The Common Warehouse metamodel (CWM) specifies interfaces that can be used to enable easy interchange of warehouse and business intelligence metadata between warehouse tools, warehouse platforms and warehouse metadata repositories in distributed heterogeneous environments. CWM is the standard for data repository integration. It standardizes how to represent schema, schema transformation models and data mining tools. In addition, CWM models enable users to trace the lineage of data, CWM provides objects that describe where the data came from and when and how the data was created. Instances of the metamodel are exchanged via XML Metadata Interchange (XMI) documents. Initially CWM contained a local definition for a data translation facility [17].

7.2 The basic concepts

The concepts that are most important in MDA are platform independent model (PIM), platform specific model (PSM) and the computation independent model (CIM). PIMs and PSMs can be understood to be the views from different viewpoints with different degree of dependence of the platform.

7.2.1 Model

A model of a system is a description or specification of that system and its environment for some certain purpose. A model is often presented as a combination of drawings and text. The text may be in a modelling language or in a natural language [3].

The model is a formal specification of the function, structure and behaviour of the system within a given context and from a specific point of view. A model is often represented using a formal notation (such as UML) extended with natural language expressions where appropriate. A formal specification is based on the language with well defined semantics associated with each of its constructs. This differentiates the formal description from the simple diagrams including some boxes and lines.

The term model-driven describes an approach to software development, where models are used as the primary source for documenting, analyzing, designing, constructing, deploying and maintaining a system.

7.2.2 Meta modelling architecture

7.2.2.1 Model layers

The MDA includes a meta model framework, referred to as a layered meta modelling architecture, in which the MDA model artefacts are related on different levels of instantiation.

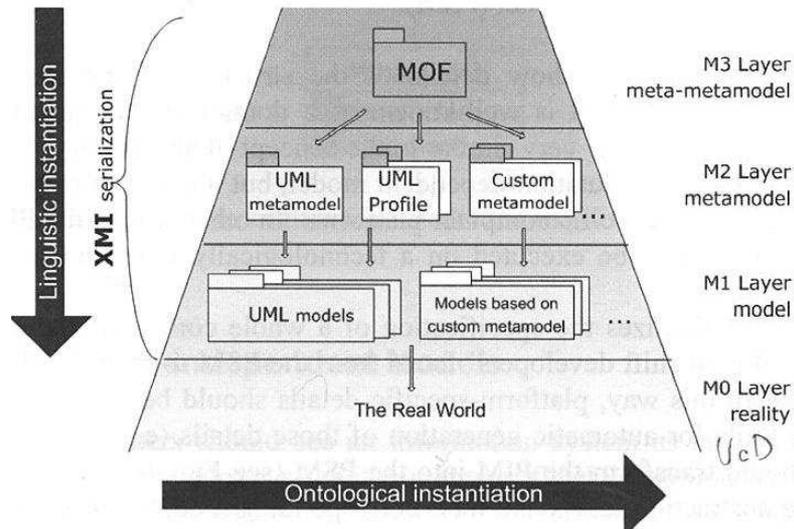


Figure 6 The MDA layered model framework (from [19])

In this framework the MOF resides in the upper most layer, being the basis for defining other MDA (meta) models on the next layer (M2). Examples of MDA meta models are the previously mentioned MDA standards, UML and CWM, as well as any other model which can be defined using the MOF. A specific application of the UML to some domain is then considered to reside on the model layer (M1). The models and model elements on each layer can be serialized into XML and communicated, using the XMI, e.g., to provide design tool interoperability.

7.2.2.2 Extension

A main purpose of this structuring of (meta) models is to provide a framework in which formalisms and mechanisms can be developed for the extensions of meta models as well as for the creation of new modelling languages, including also model transformations and mappings. Ultimately, these functions should be amendable for (semi) automated support in systems design tools.

The UML provides extension mechanisms on the meta model layer, in terms of so called profiles. A (UML) profile is a set of extensions of the basic model elements of the UML for some specific purpose or domain (extensions are based on built-in extension primitives, e.g. stereotypes for class refinement). Examples of some existing profiles include; a profile for modelling based on XML Schema, a UML profile for Enterprise Application Integration, and profiles for web applications.

7.2.2.3 MDA and ontology

Recognizing the growing importance attributed to the application of ontologies, in the semantic web field and in service oriented computing in general, efforts are underway to integrate ontology modelling into the MDA framework. Several UML profiles have also been proposed in order to use the UML for ontology design, including support in existing design tools. The Ontology Definition Metamodel (ODM) is an example of the recent efforts within the OMG to provide MDA support for ontology development [19]. This work is based on the RDF-S and OWL metamodels, and has also suggested a specific UML ontology profile as well as mappings to other meta models, as part of the overall MDA ontology framework.

7.2.3 Architecture

The architecture of a system is a specification of the parts and connectors of the system and the rules for the interactions of the parts using the connectors. In the MDA context, the parts, connectors and rules are expressed as the set of interrelated models.

7.2.4 Platform

In accordance to [3], a platform can be viewed as a coherent set of interfaces and specified usage patterns, which may or may not be implemented by a set of subsystems and technologies. What counts as a platform is relative to the purpose of the modeller. Client of a platform make use of it without concern of for its implementation details. For many MDA users, middleware is a platform, for a middleware developer an operating system is the platform. Thus a platform-independent model of middleware might appear to be a highly platform-specific model from the point of view of an application developer [3]. Examples of the platforms include operating systems, programming languages, databases, user interfaces, middleware solutions, etc.

7.2.5 Platform independence

Generally, platform independent model is a model that is not dependent on any platform. Since every model depends at least on the modelling language (which is the platform), the platform independence should not exist. In accordance to [3], the platform independence is defined as a quality, which a model may exhibit. This is the quality that the model is independent of the features of a platform of any particular type. Like most qualities, platform independence is a matter of degree. Independence is a relative indicator in the terms of measuring degree of abstraction, which separates one platform from another (i.e. one platform is either more or less abstract compared to another).

Platform independence can be viewed in two general contexts:

1. Platform independence is a property of a software system that means that the system can be used with multiple platforms of similar type (operating systems, programming languages, etc.). In this context, the software system is an application, a component or a standard.
2. Platform independence is a quality that can be exhibited by the model to some degree. The model has a degree of independence from any platform from a given class of platforms (model is independent from operating system, since it does not have references to any type of operating system).

An important aspect of platforms is, that models can be specified in the terms of platforms. A model is always defined in the terms of some modelling language (e.g. UML, which can be practically also viewed as a platform). A model defined in UML that models a Java program is the example of a model, which is dependent on two platforms: UML platform and Java platform. Because, the elements of Java language are expressed in UML, there must exist the mapping from UML to Java. This mapping can be formal, informal, or even mental.

A model is dependent on the platform in the following general cases: (1) when the platform is a modelling language and the model is specified using that language, or (2) when the model refers to elements of the platform.

7.2.6 Platform model

A platform model is a specification of a platform. In accordance to [3], a platform model provides a set of technical concepts that represent the different elements that make up a platform and the services provided by that platform. It also specifies the constraints on the use of these elements and services by other parts of the system.

7.2.7 Model transformation

Model transformation is the process of converting one model to another within the same system. The transformation combines platform independent model with additional information to produce the platform specific model.

7.2.8 Implementation

An implementation is a specification that provides all the information required to construct a system and to put it into operation. It must provide all of the information needed to create an object, and to allow the object to participate in providing an appropriate set of services as part of the system.

7.2.9 Views and viewpoints

A view of the system is a model of that system, that contains only that information about the system, which are relevant from the perspective of a chosen viewpoint. For the each viewpoint, there can be defined multiple views.

A viewpoint is a definition of the conventions for creating and using views. The conventions should include information about what to view and how to present and use the view.

The MDA has three basic kinds of view:

1. Computation Independent Model (CIM) is a view of a system from computation specific viewpoint. CIM focuses on the context and requirements of the system without consideration for its structure or processing.
2. Platform Independent Model (PIM) is a view of a system from platform independent viewpoint. PIM focuses on the operational capabilities a system by showing only those parts of the specification that can be abstracted from out of the specific platform (or set of platforms).
3. Platform Specific Model (PSM) is a view of a system from platform specific viewpoint. PSM includes the details relating to the use of the specific platform.

7.2.10 MDA models

MDA specifies the three general types of system models corresponding to the three MDA viewpoints.

7.2.10.1 Computation independent model

A Computation Independent Model (CIM) is often referred as a business or domain model. In accordance to [3], the CIM is defined as follows: A computation independent model is a model of a system that shows the system in the environment in which it will operate, and thus it helps in presenting exactly what the system is expected to do. The CIM represents the functional or context model.

The importance of CIM is the bridging the gap, which usually exists between domain experts and technology experts responsible for implementing the system.

In the MDA specification, the CIM requirements should be applied to the PIM and PSM constructs that implement them.

7.2.10.2 Platform independent model

Platform independent model (PIM) is a view of a system from the platform independent viewpoint. Generally, the platform independent viewpoint focuses on the aspects of a system that are independent of a given class of platforms. In accordance to [3], the platform independent viewpoint focuses on the operation of a system while hiding the details necessary for a particular platform. A platform independent viewpoint shows that part of the complete specification that does not change from one platform to another.

A PIM exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms of similar type. This is usually achieved by defining the set of services in a way that abstracts out the technical details. Other models then specify a realisation of these services in a platform specific manner.

7.2.10.3 Platform specific model

Platform specific model is a view of a system from the platform specific viewpoint. Generally, the platform specific viewpoint focuses on aspects of a system that are dependent of a given class of platforms. The platform specific viewpoint combines the platform independent viewpoint with an additional focus on the detail of the use of a specific platform by a system [3].

A PSM combines the specifications in the PIM with the details that specify how that system uses a particular platform. The PSM is relative to PIM, and PSM depends on a platform, corresponding PIM of which is not dependent on.

7.3 The MDA Process

In accordance with [6], the process of MDA approach is composed from three general steps (see Figure 7):

1. First step is design and development of the abstract application model, which is independent of any implementation platform (so called PIM).
2. Next step is the transformation of PIM to one or more platform specific models (PSMs). Created PSMs are the forms of PIM constructed in the detailed, platform depended models designed in terms of particular platform (such as .NET, EJB, etc.)
3. The last major step is the transformation of PSMs into the code. Particular PSMs contain the required information about the translation of the model components into the target code.

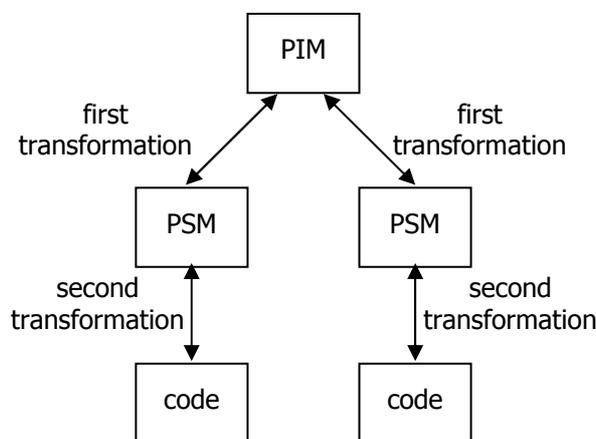


Figure 7 The general steps of the MDA process (from [6])

7.3.1 Analysis and modelling (PIM)

Systems development using MDA approach starts with the creation of platform independent model (PIM). This model is the high abstraction that is independent from any implementation technology. At this level, there is the need for some modelling language capable to describe all required facts about the application. PIM is usually defined using Unified Modelling Language (UML), but there can be used also other notations when appropriate. Behaviour and constraints can be defined using a formal notation (UML models) or informal notations (natural language) as appropriate.

The model usually has multiple levels of PIMs. The base PIM defines only the business functionality and behaviour. The design of base PIM requires cooperation of modelling and business experts. The base model expresses the business rules and functionality as much as possible independent of any

implementation platform. Models at the next level define some aspects of technology, platform specific details are missing (e.g. persistence, security, configuration information, etc.). This level of the model enables more precise mapping to PSM.

The PIM model, which is the result of the first development phase, specifies the functionality and the behaviour of the application. The PIM is created in the form of diagrams (usually UML). Class and object diagrams define the structure, sequence and activity diagrams describe the behaviour, class and object names with semantic notations specify the business factors, other aspects of the model incorporate platform independent properties of component structure and behaviour [7].

According to [8], the platform independent models provide two main advantages:

1. Modelling process and the functional design of the application does not take into account any platform and implementation details. This approach gives the freedom to concentrate only on the business rules.
2. Functionality description does not include any implementation details, so it should be easier to create various implementations on different platforms.

7.3.2 Modeling and design (PSM)

When the PIM is complete, it represents the input to the mapping phase, which produces one or more platform specific models (PSMs). Each platform specific model is adapted to specify the application in the terms of implementation constructs that are available in one specific implementation technology [5]. Platform specific model can be described in two ways [8]:

1. using UML diagrams (class, sequence, activity diagrams, etc.)
2. using interface definitions in a several implementation technology (XML, Java, etc.)

In both cases, the behaviour and the constraints are specified using a formal notation (UML) or informal notation (natural language) as appropriate.

According to [3], there exist two ways how to map PIM into specific PSM: (1) model type mapping and the (2) model instance mapping, which uses the additional transformation information.

Model type mapping is based on the types of the model elements. This mapping defines, how the different element types of PIM are transformed to different element types of PSM. The general view of basic mapping approaches is shown on Figure 8.

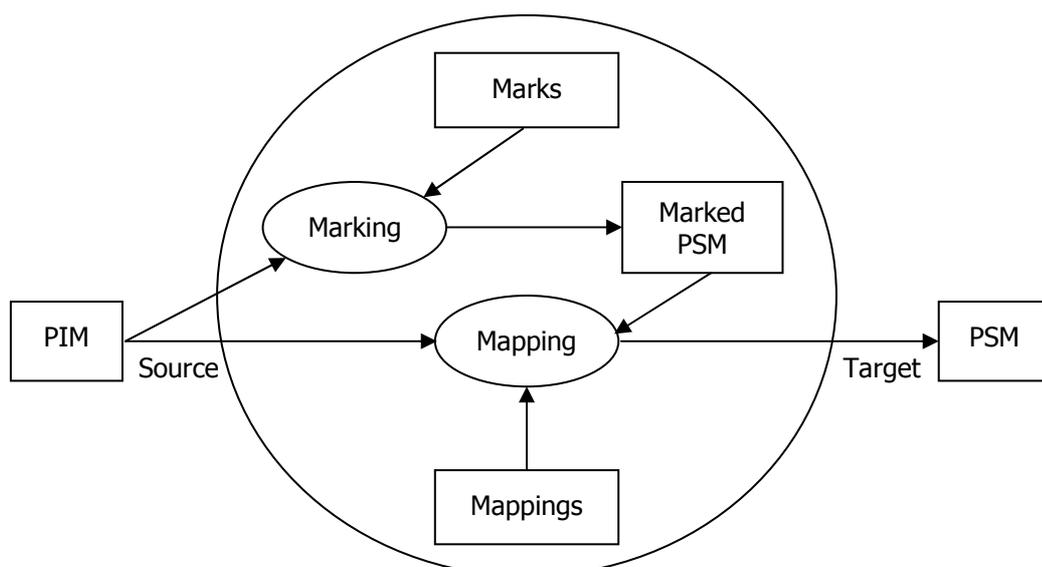


Figure 8 General types of PIM to PSM mapping (from [21])

Special case of the model mapping is metamodel mapping, where types of model elements in the PIM and PSM are specified as Meta Object Facility (MOF) metamodels. The mapping contains the

rules or algorithms which for all instances of types in the metamodel specifying PIM language generate the instances of types in the metamodel of PSM language(s).

The types available to model the PSM (or even the PIM) may not be specified as MOF metamodel. In this case, the mappings can be defined as transformations of instances of types in PIM into instances of types in PSM expressed in other languages, including natural language.

Model instance mapping uses the marks. This approach requires identification of elements in PIM, which should be transformed in a particular way independent of the target PSM platform. Marks represent the concepts of PSM and the elements of PIM are marked to indicate, how to transform them. An element of PIM may be marked several times with marks of various different mappings and then transformed in accordance of the each of the mappings. The marking of PIM element usually defines the use of specific rules for transformation or conversion of an element. The result of the marking process is so called Marked PIM, which is transformed into PSM using mapping.

The both of the mapping approaches can be combined. Model type mapping specifies the rules, how to transform the elements of PIM types to elements of PSM types. However, without the ability of using additional information (marking) for use by transformation, the mapping will be deterministic and dependent only on platform independent information. Marks represent the additional information, used by transformation tools, that is not appropriate to be presented in the model semantics.

The marks can come from different sources [3]. These include:

- types from a model, specified by classes, associations, or other model elements
- roles from a model, for example, from patterns
- stereotypes from a UML profile
- elements from a MOF model
- model elements specified by any metamodel

Marks can be also used as a specification of requirements on the implementation. Instead of addressing the target of transformation, mark defines the requirements on the target. Transformation then chooses the target that is appropriate for specified requirements. Some types of marks need to be structured, constrained or modelled. In the case, when the set of marks specifies the alternative mappings for the concept, this marks need to be grouped. This set represents the mapping alternatives, especially in the case, when only one of the marks can be applied to the particular model element.

It is also possible to define the templates for the mappings. The template is the parameterized model that defines the particular kinds of transformations and may contain the more precise specifications of the transformation. Templates can be used in the transformation rules of the type mapping or the set of marks can be associated with the particular template.

Finally, the transformation of PIM into PSM can be realised manually, with the computer assistance or automatically. Transformation process is the conversion of one model to another model of the same system. Input of the transformation is the PIM (or marked PIM) and the mapping (see Figure 8). The result is the PSM and the record of transformation. The transformation record contains the map from the PIM elements to corresponding PSM elements and the information describing which parts of mapping were used for particular parts of transformation.

Some transformation tools may transform the PIM directly into the required code. Such an approach may also produce the PSM as the utility for understanding or debugging the generated code.

7.3.3 Implementation and deployment

The last phase of MDA process takes PSMs as input and generates (parts of) the implementation code, and other kinds of supporting artefacts such as configuration files, component descriptor files, deployment files, scripts, etc.. Ideally, the ability to generate a complete implementation, the need for writing the code by hand is replaced by automated process. Unfortunately, even the most

advanced MDA tools are not able to fulfil this goal entirely [18]. Much code can be generated from a functionally complete PSM, but there is still significant amount of code, which have to be added manually in order to produce a complete, executable and deployable component or application [19].

7.4 Transformations

The concept of transformations has a very important role in the MDA process. Generally, the transformation in the MDA is the automatic process of converting one artefact into another artefact, using a tool, provided rules that describe, how to realise the transformation. A model that is independent of a given platform (PIM) is transformed, using some extra information, to a new model that is dependent of the platform (PSM) (see Figure 9).

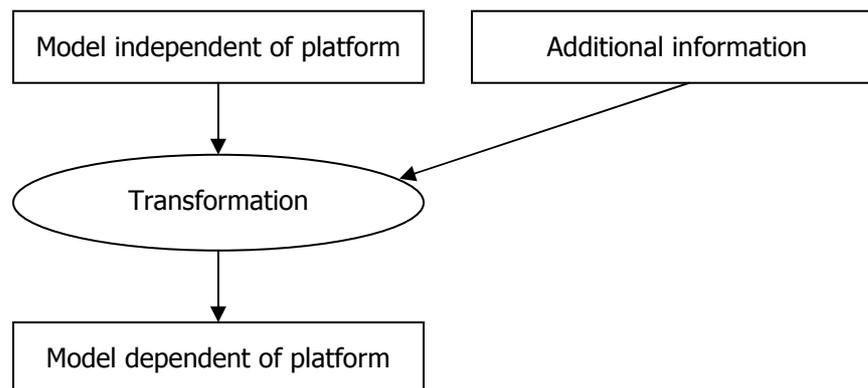


Figure 9 General view of a MDA transformation

In accordance to [5], the transformation, transformation definition, transformation rule and transformation tool are specific, different entities. These concepts can be further elaborated as follows:

1. Transformation definition is a collection of rules, that specify, how a model in the source language can be converted into a model in a source language. The transformation definitions are used by transformation tools to generate the transformations.
2. Transformation rule is a definition, how one ore more constructs in a source language can be converted into one ore more constructs in a target language.
3. Transformation tool performs a transformation for the specific source model according to a transformation definition. Selection of good tool is very important in a whole process. The transformation tool should have the following characteristics:
 - a) The tool should support different platforms.
 - b) The tool should provide the flexibility to switch within one platform between different implementation strategies, application architectures and coding patterns.
 - c) The tool should support different modelling languages and transformation definitions.
 - d) The tool should support standard domain specific models or blueprints.
 - e) The tool should be able to integrate with other automatic software development and maintenance tools.

There are three general approaches to model transformation [3]:

1. A first approach is the definition of a mapping from the metamodel of PIM to the metamodel of PSM. This approach is practically the same process as the translation of some computer language to another.
2. Second approach is to prepare the PIM model before it is transformed, by using the marks. Marks provide extra information of modelled system, they should not be included in the PIM,

because they can be specific with the PSM model created by the mapping. The marks can be used in combination with any other transformation approach.

3. Third approach is to create PIM that contains classes that are the subclasses of more generic super-classes. A mapping is realised as a transformation of the super-classes to the versions of classes that are specific to the required target platform.

A special type of transformation is required, when two or more models have to be merged to create the model, which contains the information from merged models. Merging is needed, when there are multiple models describing various aspects of the same system. The merging process should be trivial in cases, when multiple models describe different parts of the system. On the other side, merging procedure is not trivial in case, when multiple models specify the various aspects of the same part of the system. Mentioned approaches can be combined.

Transformations can be realised manually, automatically or semi-automatically:

- Manual transformation is, practically, the conventional approach of software design. Creating the implementation code while looking at the model can be considered as a manual transformation.
- A case of semi-automatic transformation is a transformation, where a PIM is extended with marks and where the tool uses the marks to guide the transformation.
- Automatic transformation is a transformation, where tool does not need any additional information to transform a model. In accordance to [3], in such a context, it is possible for an application developer to build a PIM that is complete as to classification, structure, invariants, and pre- and post-conditions. The developer can then specify the required behaviour directly in the model, using an action language. This makes the PIM computationally complete; that is, the PIM contains all the information necessary to produce computer program code.

7.5 References

- [1] <http://www.omg.org/mda/>
- [2] Lewis, Grace A. and Wraga, Lutz. Approaches to Constructive Interoperability (CMU/SEI-2004-TR-020 ESC-TR-2004-020). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.
- [3] Object Management Group. MDA Guide Version 1.0.1., 2003.
- [4] Object Management Group. MDA: Executive Overview. 2005.
- [5] Kleppe A, Warmer J, Bast W, MDA Explained - The Model Driven Architecture: Practice and Promise, Addison-Wesley, 2003.
- [6] Klasse Objecten, www.klasse.nl/mda/mda-introduction.html, Klasse Objecten Inc
- [7] Siegel J, Developing in OMG's Model-Driven Architecture: White Paper Revision 2.6, Object Management Group, November 2001.
- [8] Model Driven Architecture: A Technical Perspective, Architecture Board MDA Drafting Team, Document Number ab/2001-02-04, xml.coverpages.org/OMG-tech-01-02-04.pdf, Object Management Group, February 2001.
- [9] UML overview, <http://www.developer.com/design/article.php/1553851>
- [10] Unified Modeling Language, The official homepage of UML, www.uml.org, Object Management Group.
- [11] Meta Object Facility, Document – formal/02-04-03, www.omg.org/docs/formal/02-04-03.pdf, Object Management Group, 2003.
- [12] MOF overview, <http://www.omg.org/technology/documents/formal/mof.htm>
- [13] OMG's Meta-Object Facility Home Page, <http://www.omg.org/mof/>

- [14] XML Metadata Interchange, Specification document v2.0 – formal/03-05-02, www.omg.org/docs/formal/03-05-02.pdf, 2003.
- [15] Common Warehouse Metamodel, Specification v1.1, Document – formal/2003-03-02, www.omg.org/docs/formal/03-03-02.pdf, Object Management Group, 2003.
- [16] <http://www.omg.org/technology/documents/formal/xmi.htm>
- [17] <http://www.omg.org/technology/documents/formal/cwm.htm>
- [18] Parr, S. & Keith-Magee, R. The Next Step Applying the Model Driven Architecture to HLA In Proceedings of Spring Simulation Interoperability Workshop. Workshop paper: 03S-SIW-123, 2003.
- [19] Pokorny, T., The Model Driven Architecture: Not Easy Answers, SimTecT, 2005.
- [20] Gasevic D., Djuric D. Devedzic V., Model Driven Architecture and Ontology Development, Springer, 2006.
- [21] Alhir S, Understanding the Model Driven Architecture, Methods & Tools: An international software engineering digital newsletter published by Martinig & Associates, 2003.

8 Grid technologies

8.1 Introduction

The term "Grid" was coined to denote a proposed distributed "cyber-infrastructure" for advanced science and engineering. The term is now understood to refer to technologies and infrastructure that enable *coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations* [1]. This sharing relates primarily to direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource-brokering strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals or institutions defined by such sharing rules form a *virtual organization* (VO) [2].

Grid concepts are particularly relevant to high energy physics (HEP) due to the collaborative nature of HEP experiments and the increasing complexity of data analysis tasks, and hence a need for next-generation experiments to exploit large distributed collections of shared resources. The broad significance of Grid concepts, in advanced scientific collaborations and in business, means that HEP is just one of a number of communities (particularly ambitious and important one) that are developing or driving Grid technologies. The resulting interrelationships make it important to understand the state of the art and likely future directions in this field.

8.2 Grid Architecture

Grid technologies comprise protocols, services, and tools that address the challenges that arise when we seek to build scalable VOs. These technologies include security solutions that support management of credentials and policies when computations span multiple institutions; resource management protocols and services that support secure remote access to computing and data resources and the co-allocation of multiple resources, information query protocols and services that provide configuration and status information about resources, organizations, and services; and data management services that locate and transport datasets between storage systems and applications [3]. In the *Fabric*, we have the resources that we wish to share: computers, storage systems, data, catalogues, etc. The *Connectivity* layer provides communication and authentication services needed to communicate with these resources. *Resource* protocols (and, as in each layer, associated APIs) negotiate access to individual resources. *Collective* protocols, APIs, and services are concerned with coordinating the use of multiple resources, and finally application toolkits and applications themselves are defined in terms of services of these various kinds. Other papers present views on necessary components of a Grid architecture and the additional services required within a Data Grid architecture.

Grid computing describes the linking together of distributed computational resources to provide flexible access and a common interface for the user. Meta-computing extends this concept to enable distributed systems or supercomputers to aggregate their resources to outperform the limitations of a single computing system. To achieve these goals software systems must be provided which use Internet technology, now common in e-Commerce, for the benefit of the computational science community. Distributed computing systems offer more than just a large CPU resource. A software environment of unprecedented quality and functionality is emerging along with the use of the Internet for E-commerce and leisure purposes. This is driven by a combination of the computer industry and the loose collection of worldwide "freeware" programmers. Geoffrey Fox has referred to this as the "Distributed Commodity Computing and Information System" [1]. In the USA and Japan there are several alliances of computing centres separated by large distances. In Europe, Germany has taken a lead because of the regional computing centres. In the UK the national facilities (CSAR, EPCC) and the JREI-funded centres may be (but are not yet) a source of similar resources.

The whole concept is often referred to as a "Computational Grid". Computers on a grid can solve very large problems requiring, for instance, more main memory than is available on a single

machine. The use of these systems as single computational platforms is an active area of research, however given the high latency of wide area connections and the problems of heterogeneity such use is unlikely to be very widespread for most applications. The real value of the Grid comes instead from the ability to access remote heterogeneous resources in a common way.

A key concept is that of "ownership". A Grid is a "federation" of resources that may be accessed in a transparent way by grid users [4]. This raises the fundamental question of "accounting" for resource usage, whether it is CPU time, disk, memory, licensed software or preserved data. Whilst this is perhaps the most important issue to be considered in implementing a national Grid environment we do not consider it further in this report. Instead we focus on how the scientific user might benefit from such an ideal environment.

Distributed software tools, and especially those which facilitate very complex "coupled" applications to be constructed and used are likely to be of growing interest over the coming few years. They are however difficult to implement, and it is more likely that data management or throughput services will be more common in the short term. There is however already a very wide range of packages both commercial and public domain that are relevant to Grid computing. We outline the five most significant here: GLOBUS, STA, LSF, UNICORE, Legion and Jini.

This section briefly highlights some of the general principles that underlie the construction of the grid. In particular, the idealized design features that are required by a grid to provide users with a seamless computing environment are discussed. There are three main issues that characterize computational grids:

- **Heterogeneity:** a grid involves a multiplicity of resources that are heterogeneous in nature and might span numerous administrative domains across wide geographical distances.
- **Scalability:** a grid might grow from few resources to millions. This raises the problem of potential performance degradation as a Grids size increases. Consequently, applications that require a large number of geographically located resources must be designed to be extremely latency tolerant.
- **Dynamicity or Adaptability:** in a grid, a resource failure is the rule, not the exception. In fact, with so many resources in a Grid, the probability of some resource failing is naturally high. The resource managers or applications must tailor their behaviour dynamically so as to extract the maximum performance from the available resources and services.

The steps necessary to realize a computational grid include:

- **Grid Fabric:** It comprises all the resources geographically distributed (across the globe) and accessible from anywhere on the Internet. They could be computers (such as PCs or Workstations running operating systems such as UNIX or NT), clusters (running cluster operating systems or resource management systems such as LSF, Condor or PBS), storage devices, databases, and special scientific instruments such as a radio telescope.
- **Grid Middleware:** It offers core services such as remote process management, co-allocation of resources, storage access, information (registry), security, authentication, and Quality of Service (QoS) such as resource reservation and trading.
- **Grid Development Environments and Tools:** These offer high-level services that allows programmers to develop applications and brokers that act as user agents that can manage or schedule computations across global resources.
- **Grid Applications and Portals:** They are developed using grid-enabled languages such as HPC++, and message-passing systems such as MPI. Applications, such as parameter simulations and grand-challenge problems often require considerable computational power, require access to remote data sets, and may need to interact with scientific instruments. Grid portals offer web-enabled application services — i.e., users can submit and collect results for their jobs on remote resources through a web interface. In attempting to facilitate the collaboration of multiple organizations running diverse autonomous heterogeneous resources, a number of basic principles should be followed so that the grid environment:

- Does not interfere with the existing site administration or autonomy;
- Does not compromise existing security of users or remote sites;
- Does not need to replace existing operating systems, network protocols, or services;
- Allows remote sites to join or leave the environment whenever they choose;
- Does not mandate the programming paradigms, languages, tools, or libraries that a user wants;
- Provides a reliable and fault tolerance infrastructure with no single point of failure;
- Provides support for heterogeneous components;
- Uses standards, and existing technologies, and is able to interact with legacy applications;
- Provides appropriate synchronization and component program linkage

8.2.1 GLOBUS

The Globus Toolkit is a community-based, open-architecture, open-source set of services and software libraries that support Grids and Grid applications [5]. The Toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection, and portability. It is packaged as a set of components that can be used either independently or together to develop useful Grid applications and programming tools. Globus Toolkit components include the Grid Security Infrastructure (GSI), which provides a single-sign-on, run-anywhere authentication service, with support for delegation of credentials to sub-computations, local control over authorization, and mapping from global to local user identities; the Grid Resource Access and Management (GRAM) protocol and service, which provides remote resource allocation and process creation, monitoring, and management services; the Metacomputing Directory Service (MDS), an extensible Grid information service that provides a uniform framework for discovering and accessing system configuration and status information such as compute server configuration, network status, or the locations of replicated datasets. Data Grid-specific technologies include a replica catalogue, GridFTP, a high-speed data movement protocol, and reliable replica management tools. For each of these components, the Toolkit both defines protocols and APIs and provides open source reference implementations in C and, in most cases, Java. A variety of higher-level services can be, and have been, implemented in terms of these basic components. GLOBUS is probably the largest current academic project. It involves joint work of Argonne National Laboratory and the University of Southern California's Information Science Institute with many additional contributors. Researchers are developing a basic software infrastructure for computations that integrate geographically distributed computational and information resources. The Globus Grid programming toolkit is designed to help application developers and tool builders overcome the challenges in the construction of "Grid-aware" scientific and engineering applications. It does so by providing a set of standard services for user authentication, resource location, resource allocation, configuration, communication, file access, fault detection, and executable management. These services can be incorporated into applications and/ or programming tools in a mix-and-match fashion to provide access to needed capabilities.

8.2.2 Seamless Thinking Aid

Seamless Thinking Aid (STA) is a Web-aware Java-based environment which includes a number of tools to assist parallel programming. The goal is to allow larger calculations and to couple applications with different memory or architectural requirements.

8.2.3 LSF

Load Sharing Facility (LSF) is a product of Platform Computing, widely used for corporate computational resource management, especially in the engineering industry. Platform aims to provide the best application resource management solutions for enterprise, allowing administrators

to intelligently harness and leverage the maximum power from their existing computing systems by using idle cycles in a flexible and dynamic manner. The system has a broad range of academic and commercial users. As well as monitoring load information such as, CPU queue length and utilisation, available user memory, paging and disk I/O rate, etc. LSF provides facilities to transfer work between locally managed or remote systems, e.g. to access machines with particular software licenses. This can work over autonomous and widely separated sites. Platform Computing is pioneering an open distributed resource management initiative with a number of other partners.

8.2.4 UNICORE

Uniform Access to Computing Resources was originally a project funded by the German Federal Republic to connect together several important regional super-computing centres [6]. The strong federal political structure of the DBR makes this grid-based model particularly relevant and provides a grid environment that is also a very suitable model for a grid connecting the supercomputing centres of the whole EU. UNICORE lets the user compose and edit structured jobs with a graphical job preparation client on a local workstation or PC. Jobs can be submitted to run on any platform in the UNICORE grid, and the user can monitor and control the submitted jobs through the job monitor client.

8.2.5 Legion

Legion is an integrated grid-computing system that, like Globus, has been deployed at a number of sites in the USA [7]. It arose from an object-based software project at the University of Virginia beginning in 1993. Legion supports existing codes written in MPI and PVM, as well as "legacy" binaries. Key capabilities include:

- Eliminating the need to move and install binaries manually on multiple platforms;
- Providing a shared, secure virtual file system that spans all the machines in the system;
- Providing strong PKI-based authentication and flexible access control for user objects;
- Supporting remote execution of legacy codes, and their use in parameter space studies.

Legion is the second grid project that has been adopted by the US NSF at its National Partnership for Advanced Computational Infrastructure (NPACI) sites. NASA and the DoD are also running Legion test beds.

8.2.6 Jini

Jini is a Java middleware technology designed to support the general requirements of federating network resources [8]. Whilst Jini will not currently support an HPC grid it represents the natural direction for grid technology and it is likely that either Jini, or something that builds on its design concepts will be an integral part of the Grid of the future. A Jini system is a distributed system based on the idea of federating groups of users and the resources required by those users. Jini leverages the Java environment to provide systems that are far more dynamic than is currently possible in networked groups where configuring a network is a centralised function done by hand. Although Jini uses Java, a Jini Grid could support the execution of code written in an arbitrary language.

8.3 OGSi

Building on both Grid and Web services technologies, the Open Grid Services Infrastructure (OGSI) defines mechanisms for creating, managing, and exchanging information among entities called *Grid services* [5]. Succinctly, a Grid service is a Web service that conforms to a set of conventions (interfaces and behaviours) that define how a client interacts with a Grid service. These conventions, and other OGSi mechanisms associated with Grid service creation and discovery, provide for the controlled, fault-resilient, and secure management of the distributed and often long-lived state that is commonly required in advanced distributed applications. In a separate document, we have presented in detail the motivation, requirements, structure, and applications that underlie OGSi.

Here we focus on technical details, providing a full specification of the behaviours and Web Service Definition Language (WSDL) interfaces that define a Grid service.

The *Open Grid Services Architecture* (OGSA) integrates key Grid technologies (including the Globus Toolkit) with Web services mechanisms to create a distributed system framework based on the *Open Grid Services Infrastructure* (OGSI). A *Grid service instance* is a (potentially transient) service that conforms to a set of conventions, expressed as Web Service Definition Language (WSDL) interfaces, extensions, and behaviours, for such purposes as lifetime management, discovery of characteristics, and notification. Grid services provide for the controlled management of the distributed and often long-lived state that is commonly required in sophisticated distributed applications. OGSI also introduces standard factory and registration interfaces for creating and discovering Grid services. OGSI version 1.0 defines a component model that extends WSDL and XML Schema definition to incorporate the concepts of

- stateful Web services,
- extension of Web services interfaces,
- asynchronous notification of state change,
- references to instances of services,
- collections of service instances, and
- service state data that augments the constraint capabilities of XML Schema definition.

In this specification we define the minimal, integrated set of extensions and interfaces necessary to support definition of the services that will compose OGSA. No specification is written in isolation, and Web services and XML are particularly dynamic and evolving environments. We intend to ensure that the evolution of OGSI conforms with broader standards that evolve. Many of the concepts are defined – for example, *serviceData* (§6) – are special cases of more general concepts that may appear in XML documents, messages, and Web services. In addition, we anticipate that work to implement the OGSI Web services component model in various hosting environments, such as J2EE, will lead to the need for modifications to subsequent revisions of this OGSI V1.0 specification.

OGSI adds these values to the Web Services:

- The OGSI Specification defines a subset of the behaviours of web services that are relevant to grid computing. In a sense, the OGSI Specification and the OGSI Working Group are like the Web Services Interoperability Organization (WS-I), but concentrating on the standardization necessary to make truly large-scale grids possible. For example, OGSI defines a lifetime management interface for transient grid service instances.
- OGSI defines the idea of transient, i.e. short-lived, services. For example, a computational job could be viewed as a service. Currently, web services do not support this notion.
- The OGSI Specification defines a two-level naming scheme based on Grid Service Handles (GSHs) and Grid Service References (GSRs). Each GSH is a global identifier for a unique grid service instance for all time. While a GSH is global unique handle for a grid service, it does not contain all the (possibly) dynamic information needed to communicate with a client. OGSI provides for a GSH to be resolved into one or more GSRs which contain all the information needed to communicate with clients using one or more protocol bindings.
- OGSI provides a model for accessing the internal state that a grid service chooses to publicly expose. The Service Data Element (SDE) model provides standard mechanisms for querying, updating and adding and removing data associated with each grid service instance.

8.3.1 WSRF (Web Services Resource Framework)

OGSI is now obsolete, and has been superseded (in practical terms) by WSRF [9]. Web services groups started to integrate their own approaches to capturing state into the Web Services Resource Framework (WSRF). With the release of GT4 (latest version of the Globus Toolkit), the open source

tool kit is migrating back to a pure Web services implementation (rather than OGSI), via integration of the WSRF.

WSRF is a set of Web service specifications being developed by the OASIS organization. Taken together and with the WS-Notification (WSN) specification, these specifications describe how to implement OGSA capabilities using Web services. The Globus Toolkit 4.0 and later versions provide an open source WSRF development kit and a set of WSRF services.

8.3.1.1 Motivation

Web services must often provide their users with the ability to access and manipulate state, i.e., data values that persist across, and evolve as a result of, Web service interactions [5]. And while Web services successfully implement applications that manage state today, we need to define conventions for managing state so that applications discover, inspect, and interact with state resources in standard and interoperable ways. The WS-Resource Framework defines these conventions and does so within the context of established Web services standards.

8.3.1.2 Origins

Initial work on the WS-Resource Framework has been performed by the Globus Alliance and IBM, who released initial architecture and specification documents with co-authors from HP, SAP, Akamai, TIBCO and Sonic (see below) for public comment and review on January 20, 2004. These documents were submitted to the OASIS standards group in March 2004. The WSRF Technical Committee was formed to work on WS-ResourceProperties, WS-ResourceLifetime, WS-ServiceGroup, and WS-BaseFaults specifications. The WSN Technical Committee was formed to work on WS-BaseNotification, WS-Topics, and WS-BrokeredNotification specifications.

The WS-Resource Framework is inspired by the work of the Global Grid Forum's Open Grid Services Infrastructure (OGSI) Working Group. Indeed, it can be viewed as a straightforward refactoring of the concepts and interfaces developed in the OGSI V1.0 specification in a manner that exploits recent developments in Web services architecture (e.g., WS-Addressing).

8.4 References

- [1] F. Berman, G. Fox, A.J.G. Hey (Eds.): Grid Computing: Making the Global Infrastructure a Reality, Wiley, 2003.
- [2] J. Joseph and C. Fellenstein: Grid Computing, Prentice Hall/IBM Press, 2004.
- [3] I. Foster, C Kesselman: Computational Grids, The Grid – Blueprint for a new Computing Infrastructure, Morgan Kaufmann, 1999.
- [4] I. Foster and C. Kesselman (Eds): The Grid 2, Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 2004.
- [5] The Globus Alliance, <http://www.globus.org>
- [6] Unicore, <http://www.unicore.org>
- [7] Legion, <http://legion.virginia.edu>
- [8] Jini, <http://sun.com/jini>
- [9] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf
- [10] <http://www.oasis-open.org/>

9 Wireless networks and devices

9.1 Wireless Networks

This section describes the most promising and widely available wireless network communication protocols with an emphasis on short distance communication.

9.1.1 Bluetooth

Bluetooth (IEEE 802.15.1) is a low cost, mature and safe technology. It was first introduced in 1994 by Ericsson as a way to connect mobile phones with accessories. Five companies formed the Bluetooth Special Interest Group in 1998 (Ericsson, Intel, IBM, Toshiba and Nokia) and the first specification was drafted in 1999. The first retail products were marketed in 2001, the Bluetooth specification 1.2 was published in 2003 and Bluetooth 2.0 was released in 2005 [1].

Devices are categorized into three different classes by the power they use. A class 3 device has a 1 mW transmission power and a range of 0.1-10 meters. A class 2 device has a transmission power of 1-2.5 mW and a 10-meter range. A class 1 device has a transmission power up to 100 mW and a range up to 100 meters.

A comparison of Bluetooth with other wireless technologies shows the following advantages:

- In Q3 2003, total Bluetooth product shipments worldwide exceeded 1 million units per week. In Q4 2005 there are 9 million Bluetooth chips sold every week.
- Bluetooth devices can be configured either as master (initiating the connection) or as slaves (waiting for another device to request communication).
- Its power consumption is reasonably low for a wide variety of applications (Less than 40 mA in emission TX mode and around 20 μ A in standby or sleep mode) and still decreasing with new Bluetooth 2.0 specifications
- Bluetooth allows "piconets" in which up to 7 devices can be simultaneously connected to a master. Piconets can be interconnected between themselves and form so called "scatternets".
- In September 2003, the FDA approved the first Bluetooth system for medical purposes.
- Wi-Fi or Wireless LAN is used mostly for building high rate-high bandwidth stationary links. It is usually meant as a replacement for Ethernet. Many applications such as Voice over IP, location based applications, image transfer etc. The typical modules are large and cost about 3 times as much as a Bluetooth radio. Wi-Fi has too big a power consumption (typically 200 – 400 mA in TX mode and 20 mA in standby) to be compatible with long term recording or transmission.
- ZigBee (IEEE 802.15.4) is not yet mainstream: there are no ZigBee access points or phones available and it is prone to be suffering from interference in environments in which there are other 2.4 GHz transmissions such as Wi-Fi (for example in hospitals). It is also far from being standardized.

9.1.2 ZigBee

The ZigBee Alliance [2] is an association of companies working together to develop standards (and products) for reliable, cost-effective, low-power wireless networking and it is foreseen that ZigBee technology will be embedded in a wide range of products and applications across consumer, commercial, industrial and government markets worldwide. ZigBee builds upon the IEEE 802.15.4 standard which defines the physical and MAC layers for low cost, low rate personal area networks. It defines the network layer specifications, handling star and peer-to-peer network topologies, and

provides a framework for application programming in the application layer. The following subsections give more details on the IEEE standard and the ZigBee standard.

9.1.2.1 IEEE 802.15.4 Standard

The IEEE 802.15.4 standard defines the characteristics of the physical and MAC layers for Low-Rate Wireless Personal Area Networks (LR-WPAN). The advantages of an LR-WPAN are ease of installation, reliable data transfer, short-range operation, extremely low cost, and a reasonable battery life, while maintaining a simple and flexible protocol stack.

9.1.2.2 The Physical Layer

The physical layer supports three frequency bands: a 2450 MHz band (with 16 channels), a 915 MHz band (with 10 channels) and an 868 MHz band (1 channel), all using the Direct Sequence Spread Spectrum (DSSS) access mode. The 2450 MHz band employs Offset Quadrature Phase Shift Keying (O-QPSK) for modulation while the 868/915 MHz bands rely on Binary Phase Shift Keying (BPSK). Table 3-1 summarizes the main features of the three bands. Besides radio on/off operation, the physical layer supports functionalities for channel selection, link quality estimation, energy detection measurement and clear channel assessment to assist the channel selection.

9.1.2.3 The MAC Layer

The MAC layer defines two types of nodes: Reduced Function Devices (RFDs) and Full Function Devices (FFDs) [3]. FFDs are equipped with a full set of MAC layer functions, which enables them to act as a network coordinator or a network end-device. When acting as a network coordinator, FFDs will have the ability to send beacon, offering synchronisation, communication and network join services. RFDs can only act as end-devices and are equipped with sensors/actuators like transducers, light switches, lamps, etc. and may only interact with a single FFD. Two main types of network topology are considered in IEEE802.15.4, namely, the star topology and the peer-to-peer topology. In the star topology, a master-slave network model is adopted where the master is denoted the PAN coordinator and only a FFD can take up this role; slaves can be RFDs or FFDs and will only communicate with the PAN coordinator. In the peer-to-peer topology, a FFD can talk to other FFDs within its radio range and can relay messages to other FFDs outside of its radio coverage through an intermediate FFD, forming a multihop network. A PAN coordinator is selected to administer the multihop network operation. The PAN coordinator may operate its PAN with an upperframe or without it. In the first case it starts the superframe with a beacon serving for synchronization purposes as well as to describe the superframe structure and send control information to the PAN. The superframe is divided into an active and an inactive portion (wherein the PAN coordinator may go to sleep and save energy). The active portion is divided into fixed size slots and contains a Contention Access Period (CAP), where nodes compete for channel access using a slotted CSMA-CA protocol, and a Contention Free Period (CFP), where nodes transmit without contending for the channel in Guaranteed Time Slots (GTS) assigned and administered by the PAN coordinator. When an end-device needs to send data to a coordinator (non GTS) it must wait for the beacon to synchronize and later contend for channel access. On the other hand, communication from a coordinator to an end-device is indirect. The coordinator stores the message and announces pending delivery in the beacon frame. End-devices usually sleep most of the time and wake up periodically to see if they have to receive same messages from the coordinator by waiting for the beacon frame.

When they notice that a message is available, they request it explicitly during the CAP and the coordinator will send it. When a coordinator wishes to talk to another coordinator it must synchronize with its beacon and act as an end device.

9.1.2.4 The ZigBee Standard

ZigBee standardizes the higher layers of the protocol stack [4]. The network layer (NWK) is in charge of organizing and providing routing over a multihop network (built on top of the IEEE

802.15.4 functionalities), while the Application Layer (APL) intends to provide a framework for distributed application development and communication. The APL comprises the Application Framework, the ZigBee Device Objects (ZDO), and the Application Sub Layer (APS). The Application Framework can have up to 240 Application Objects, that is, user defined application modules which are part of a Zigbee application. The ZDO provides services that allow the APOs to discover each other and to organize into a distributed application. The APS offers an interface to data and security services to the APOs and ZDO.

9.1.2.5 The Network Layer

ZigBee identifies three device types. A ZigBee end-device corresponds to an IEEE RFD or FFD acting as a simple device. A ZigBee router is an FFD with routing capabilities. The ZigBee coordinator (one in the network) is an FFD managing the whole network. Besides the star topology (that naturally maps to the corresponding topology in IEEE 802.15.4), the ZigBee network layer also supports more complex topologies like the tree and the mesh.

9.1.2.6 The Application Layer

A ZigBee application consists of a set of Application Objects (APOs) spread over several nodes in the network [5]. An APO is a piece of software (from an application developer) that controls a hardware unit (transducer, switch, lamp) available on the device. Each APO is assigned a locally unique endpoint number that other APOs can use as an extension to the network device address to interact with it. The ZigBee Device Object (ZDO) is a special object which offers services to the APOs: it allows them to discover devices in the network and the service they implement. It also provides communication, network and security management services [6]. The Application Sublayer (APS) provides data transfer services for the APOs and the ZDO.

A ZigBee application must conform to an existing (ZigBee Alliance-accepted) application profile. An application profile defines message formats and protocols for interactions between application objects that collectively form a distributed application. The application profile framework allows different developers to independently build and sell ZigBee devices that can interoperate with each other in a given application profile. Each APO encapsulates a set of attributes (data entities representing internal state, etc.) and provides functionalities (services) for setting/retrieving values of these attributes or being notified when an attribute value changes. In the context of a profile a group of related attributes is termed a "cluster" and identified with a numeric id. Typically a cluster represents a sort of interface (or part of it) of the APO to the other APOs. The application profile must specify one of two possible communication service types. For the "Key Value Pair" (KVP) service type the ZigBee standard has predefined message layouts which must be suitably filled by APOs to request a given operation on attributes residing on a remote APO. The interactions between APOs are limited by the operations supported on attributes. The "generic message" service type is suitable for applications that do not fit in the KVP service type and leaves responsibility to the application profile for specifying message types and their contents.

9.1.3 Wi-Fi

9.1.3.1 Background

Wi-Fi is a brand originally licensed by the Wi-Fi Alliance to describe the underlying technology of wireless local area networks (WLAN) based on the IEEE 802.11 specifications. It was developed by Kyle Brown to be used for mobile computing devices, such as laptops, in LANs, etc., but is now increasingly used for more services, including Internet and VoIP phone access, gaming, and basic connectivity of consumer electronics such as televisions and DVD players, or digital cameras. More standards are in development that will allow Wi-Fi to be used by cars in highways in support of an Intelligent Transportation System to increase safety, gather statistics, and enable mobile commerce (see IEEE 802.11p)[22].

A person with a Wi-Fi enabled device such as a computer, cell phone or PDA can connect to the Internet when in proximity of an access point. The region covered by one or several access points is called a hotspot. Hotspots can range from a single room to many square miles of overlapping hotspots. Wi-Fi can also be used to create a mesh network. Both architectures are used in community networks, municipal wireless networks like Wireless Philadelphia, and metro-scale networks like M-Taipei.

Wi-Fi also allows connectivity in peer-to-peer mode, which enables devices to connect directly with each other.

9.1.3.2 Specifications

A typical Wi-Fi setup contains one or more Access Points (APs) and one or more clients. An AP broadcasts its SSID (Service Set Identifier, "Network name") via packets that are called beacons, which are usually broadcast every 100 ms. The beacons are transmitted at 1 Mbit/s, and are of relatively short duration and therefore do not have a significant effect on performance.

Since 1 Mbit/s is the lowest rate of Wi-Fi it assures that the client who receives the beacon can communicate at least 1 Mbit/s. Based on the settings, the client may decide whether to connect to an AP. If two APs of the same SSID are in range of the client, the client firmware might use signal strength to decide which of the two APs to make a connection to. The Wi-Fi standard leaves connection criteria and roaming totally open to the client. This is a strength of Wi-Fi, but also means that one wireless adapter may perform substantially better than the other. Since Wi-Fi transmits in the air, it has the same properties as a non-switched Ethernet network. Even collisions can therefore appear as in non-switched Ethernet LAN's.

9.1.3.3 Physical Layer

The IEEE (define) 802.11 standard includes a common Medium Access Control (MAC) Layer, which defines protocols that govern the operation of the wireless LAN. In addition, 802.11 comprises several alternative physical layers that specify the transmission and reception of 802.11 frames. The physical layer uses direct sequence spread spectrum (DSSS) to support operation of up to 11Mbps data rates in the 2.4GHz band.

As with other 802.11 Physical layers, 802.11b includes Physical Layer Convergence Procedure (PLCP) and Physical Medium Dependent (PMD) sub-layers. These are somewhat sophisticated terms that the standard uses to divide the major functions that occur within the Physical Layer. The PLCP prepares 802.11 frames for transmission and directs the PMD to actually transmit signals, change radio channels, receive signals, and so on.

9.1.3.4 MAC Layer

- Scanning: a radio NIC searches for access points. Passive scanning is mandatory where each NIC scans individual channels to find the best access point signal. Periodically, access points broadcast a beacon, and the radio NIC receives these beacons while scanning and takes note of the corresponding signal strengths.
- Authentication: there are two ways – open system and shared key authentication -. In an open system authentication process, a radio NIC initiates the process by sending an authentication request to the access point, and it replies with a frame containing approval or disapproval. Shared key authentication checks if the authenticating device has the correct WEP key.
- WEP: if this is the authentication method, the wireless NIC will encrypt the body of each frame before transmission using a common key.
- Fragmentation: The optional fragmentation function enables an 802.11 station to divide data packets into smaller frames. This is done to avoid needing to retransmit large frames in the presence of RF interference.

- RTS/CTS: The optional request-to send and clear-to-send (RTS/CTS) function allows the access point to control use of the medium for stations activating RTS/CTS. With most radio NICs, users can set a maximum frame length threshold whereby the radio NIC will activate RTS/CTS. For example, a frame length of 1,000 bytes will trigger RTS/CTS for all frames larger than 1,000 bytes. The use of RTS/CTS alleviates hidden node problems, that is, where two or more radio NICs can't hear each other and they are associated with the same access point.

9.1.3.5 Advantages of Wi-Fi

- Wi-Fi silicon pricing continues to come down, making Wi-Fi a very economical networking option and driving inclusion of Wi-Fi in an ever-widening array of devices.
- Wi-Fi products are widely available in the market. Different brands of access points and client network interfaces are interoperable at a basic level of service. Products designated as Wi-Fi CERTIFIED by the Wi-Fi Alliance are interoperable and include WPA2 security.
- Wi-Fi networks support roaming, in which a mobile client station such as a laptop computer can move from one access point to another as the user moves around a building or area.
- Wi-Fi is a global set of standards. Unlike cellular carriers, the same Wi-Fi client works in different countries around the world.
- Widely available in more than 250,000 public hot spots and millions of homes and corporate and university campuses worldwide.
- As of 2006, WPA and WPA2 encryption are not easily crackable if strong passwords are used.
- New protocols for Quality of Service (WMM) and power saving mechanisms (WMM Power Save) make Wi-Fi even more suitable for latency-sensitive applications (such as voice and video) and small form-factor devices.

9.1.4 HomeRF

9.1.4.1 Background

Home Radio Frequency was developed by the HomeRF Working Group, USA. The HomeRF Working Group was a consortium of mobile wireless companies that included Compaq, IBM, HP, Motorola, Proxim, and Siemens AG. The working group was disbanded in January 2004 after 802.11b networks became accessible to home users and Microsoft began including support for Bluetooth in its Windows operating systems. Thus, HomeRF became obsolete and there is currently no group developing the standard further. The archive of the HomeRF Working Group is maintained by Palo Wireless [23].

HomeRF is a Personal Area Network (PAN) standard which was compared to Bluetooth in its early days. HomeRF was specifically designed for wireless home networks while being more affordable than other wireless technologies. It is a wireless networking specification that uses Direct Sequence Spread Spectrum (DSSS) and Shared Wireless Access Protocol (SWAP) to transmit in the unlicensed 2.4 GHz frequency band. HomeRF can address up to 127 devices while achieving a maximum throughput of 10 Mbit/s within a range of 50 meters. Derived from the Digital European Cordless Telephone (DECT) standard, HomeRF used a hopping technique that changed 50 times per second with each 20 ms frame containing one Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) slot for data and six full-duplex Time Division Multiple Access (TDMA) slots for voice.

HomeRF allowed both traditional telephone signals and data signals to be exchanged over the same wireless network. Therefore, in HomeRF, cordless telephones and laptops could have shared the same bandwidth in the same home or office.

9.1.4.2 Technical Details

- Frequency hopping network: 50 hops/second
- Frequency Range: 2.4 GHz (globally available ISM band)
- Transmission power: 100 mW
- Range: typical home & yard (up to 150 meter radius)
- Data Rate: 1.6 Mbps using 4FSK modulation, 0.8 Mbps using 2FSK
- Data networking: up to 127 peer devices; technology derived from 802.11 & OpenAir
- Voice networking: 4 voice lines; technology derived from DECT (digitally enhanced cordless telephony) standard
- Compression: LZRW3-A algorithm
- Security: Hopping, 24-bit network ID, optional 56-bit encryption (one Trillion codes)

9.1.4.3 Security

HomeRF 2.0's security model is relatively transparent to the end user and very secure. HomeRF 2.0 uses a technology called frequency hopping. This keeps the 'data channel' shifting from one frequency to another many times a second. Frequency hopping makes it very hard for someone to eavesdrop on your network. Also, HomeRF 2.0 has introduced the concept of a 'network password' needed to join your network.

As well as an independent network IP, data is sent with a 56-bit encryption algorithm. The 56-bit encryption algorithm is more tamper proof than the 40-bit encryption codes previously recommended by the National Security Agency. The encryption algorithm, which was devised by security experts at Intel, is significantly stronger than the A5 algorithm used in GSM, yet it is only slightly more complex in hardware. When exporting HomeRF to countries of concern to the NSA the encryption algorithm is flexible enough to revert back to 40-bits. SWAP also makes use of LZRW3-A algorithm when compressing data.

9.1.4.4 Interference Dealing

802.11b, or wireless Ethernet, is subject to interference from 2.4GHz devices like some cordless phones. HomeRF also uses 2.4GHz but will track the particular kinds of interference in your home and work around them. It does this by figuring out what 'data channel' the interference is on, and then telling the frequency hopper to not use that channel. HomeRF 2.0 does NOT interfere with Bluetooth technologies.

9.1.4.5 Support

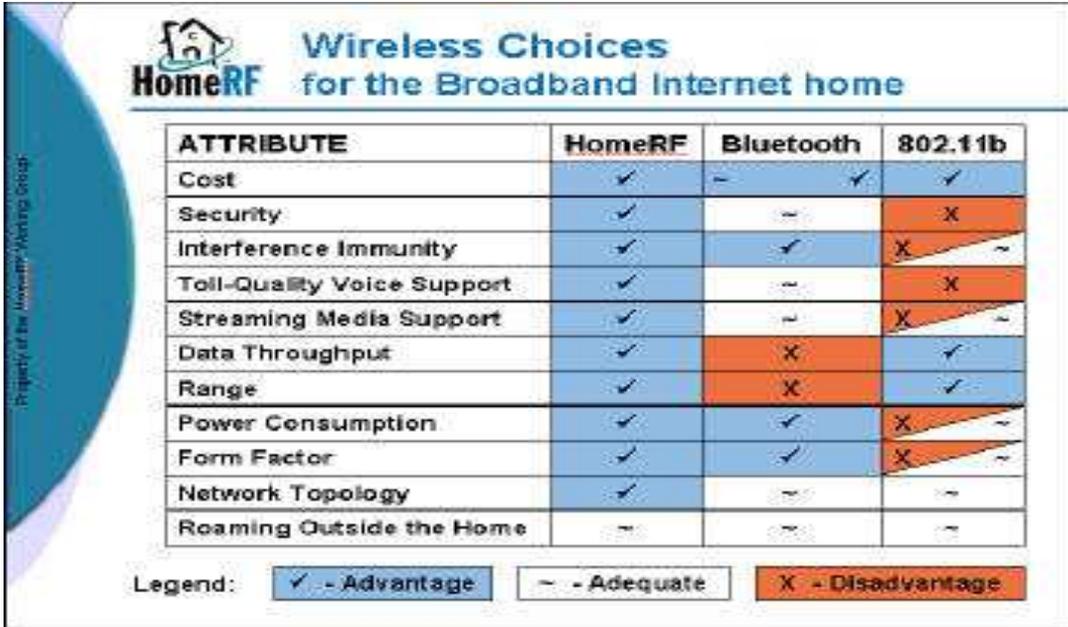
Voice communication support in HomeRF2.0 was derived from a successful European standard called DECT (Digital Enhanced Cordless Telephone). Using this technology, HomeRF 2.0 explicitly supports up to 4 simultaneous conversations and up to 8 phone handsets. In 2002, that number will be boosted to 8 simultaneous conversations.

9.1.4.6 QoS

QoS is a technology that guarantees bandwidth and prioritizes network packets. When using your network for multiple services, like voice conversations, copying files, and internet access, QoS makes sure that the important data gets across the network before less important packets. In the above example, voice conversations might be the highest priority to keep the sound quality crystal clear. QoS is an industry standard being 'added' to other networking technologies, but comes standard in HomeRF 2.0.

9.1.4.7 Power Requirements

HomeRF 2.0 was designed for more devices than laptops and computers. The HomeRF 2.0 chipset is tiny and uses very little power making it appropriate to incorporate into things like WebPads, cordless internet phones, PDA's, etc. (3.3v, 120mA Receive, 250mA transmit, 3mA standby)



The image shows a comparison table titled "Wireless Choices for the Broadband Internet home" comparing HomeRF, Bluetooth, and 802.11b across various attributes. A legend at the bottom indicates that a checkmark (✓) represents an advantage, a tilde (~) represents adequacy, and an 'X' represents a disadvantage. The HomeRF column is highlighted in blue, while Bluetooth and 802.11b columns are white with orange/red highlights for disadvantages.

ATTRIBUTE	HomeRF	Bluetooth	802.11b
Cost	✓	~	✓
Security	✓	~	X
Interference Immunity	✓	✓	X
Toll-Quality Voice Support	✓	~	X
Streaming Media Support	✓	~	X
Data Throughput	✓	X	✓
Range	✓	X	✓
Power Consumption	✓	✓	X
Form Factor	✓	✓	X
Network Topology	✓	~	~
Roaming Outside the Home	~	~	~

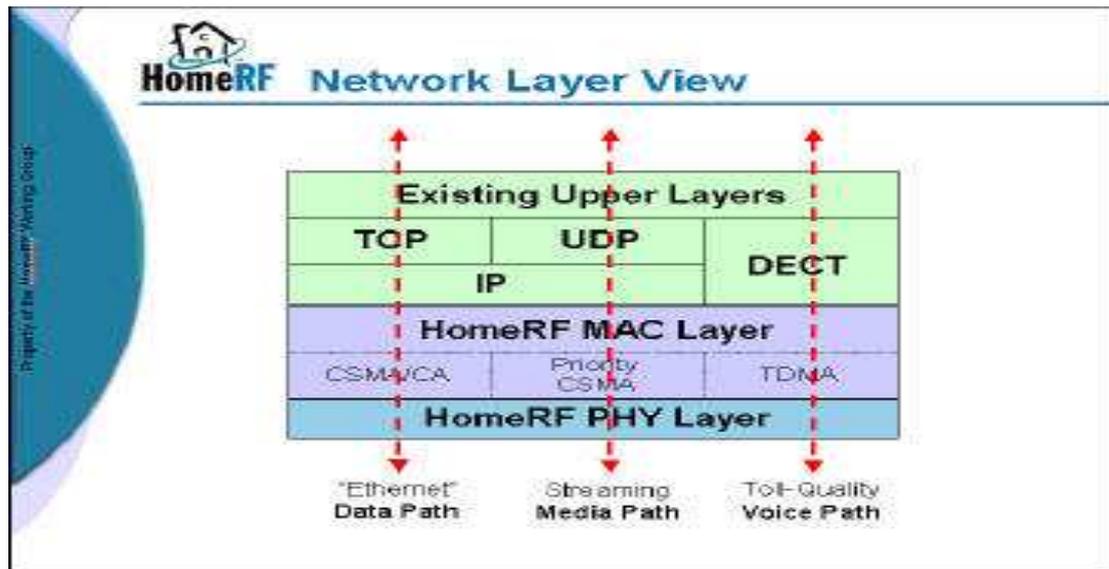
Legend: ✓ - Advantage ~ - Adequate X - Disadvantage

9.1.4.8 Physical Layer

The PHY specification for SWAP was largely adopted from IEEE802.11FH. It has been modified significantly to reduce cost, allow a single chip implementation while still maintaining more than adequate performance for home usage scenarios. Some key SWAP PHY layer specifications include: transmit power up to +24dBm, receiver sensitivity in 2FSK, optional low power transmit mode: 0 to 4 dBm for portable devices.

9.1.4.9 MAC Layer

MAC is optimized for the home environment and is designed to carry both voice and data traffic and to inter-operate with the PSTN using a subset of DECT. A TDMA service is used to support the delivery of isochronous data and a CSMA/CA service (derived from Wireless LAN standard IEEE802.11) is provided to support the delivery of asynchronous data.



9.1.5 Z-Wave

9.1.5.1 Background

Z-Wave is a wireless RF-based communications technology designed for residential and light commercial control and status reading applications such as meter reading, lighting and appliance control, HVAC, access control, intruder and fire detection, etc.

The Z-Wave technology is available in the Z-Wave Single Chip solutions. The Z-Wave protocol stack is embedded in the chips, and Flash memory is available to the manufacturer/OEM for their application software. For smooth product development, a range of manufacturing blueprints of the PCB circuitry surrounding the Z-Wave Single Chip is offered – including antenna circuitry and filters [24].

9.1.5.2 Z-Wave Features

Low Cost For Mass Market

To ensure the lowest possible cost, Z-Wave is dedicated to control and status reading applications, and therefore operates with a bandwidth of just 9.6 kbps. Z-Wave is not suited for bandwidth intensive applications such as voice/video transfer. Its bandwidth is tailored to the specific applications for which it was designed – and so is its cost per node. Innovative protocol handling techniques replace costly HW implementations to deliver the right price points. Additionally, the implementation in a mixed-signal single chip ensures the lowest cost points.

Highly Robust And Reliable

Many RF technologies communicate across the public bands. Consequently, the public bands are crowded with interference, resulting in poor reliability for most RF technologies. Z-Wave minimizes these "noise and distortion" problems by using transmission mechanisms such as 2-way acknowledgement, condensed frame formats and random back-off algorithms, ensuring highly reliable communication between all the devices in the network.

Full Home Signal Coverage

Most control systems today require physical wire connections to ensure full building coverage because the range and reliability of most wireless systems is limited. Z-Wave's dynamic routing principle, integrated into the technology, secures a virtually unlimited signal range, as each of the Z-Wave devices repeats the signal from one device to the next. The same routing principle ensures the RF-signals are routed around radio dead spots and signal reflections thereby securing a highly robust transmission covering the entire home.

Easy Network Management

Z-Wave is designed to enable automatic network address assignment at installation, simple inclusion/exclusion of nodes, and simple association/disassociation of nodes to one another. These protocol-handling techniques ensure easy installation, expansion, and management of the Z-Wave control network. Further, each Z-Wave network has its own unique Network Identifier preventing control problems or interference from neighbouring networks.

Low Power Consumption

Unlike most control systems, Z-Wave's lightweight protocol implementation and compressed frames helps keep power consumption low. Additionally, Zensys' Z-Wave single chip solutions enable advanced power saving modes for battery-operated devices such as thermostats and sensors.

Versatility

Z-Wave is a scalable protocol that was developed with the versatility to include additional features and applications as well as to connect to other protocols. To ensure future flexibility, backwards compatibility and expanded applications, Z-Wave provides multiple feature support by the use of generic command classes and a variable frame structure as well as by providing a well-defined API for OEM specific applications.

9.1.6 Wireless USB**9.1.6.1 Background**

Certified Wireless USB is the new wireless extension to USB that combines the speed and security of wired technology with the ease-of-use of wireless technology.

Certified Wireless USB will support robust high-speed wireless connectivity by utilizing the common WiMedia MB-OFDM Ultra-wideband (UWB) radio platform as developed by the WiMedia Alliance [25].

9.1.6.2 Features**Speed**

Wireless USB delivers speeds of 480Mbps at 3 meters, or approximately 10 feet, and 110Mbps at 10 meters, or approximately 30 feet. At close range, that is the same rate as Hi-Speed USB.

Power Management

Like all USB connections, Wireless USB is designed to conserve power. Sleep, Listen, Wake and Conserve modes enable users to use power only when the connection is needed.

Security

Wireless USB provides optimum data security through built-in protocols and authentication procedures, as well as encryption process during transmission.

Ease of use

Like traditional USB, Wireless USB is simple to install and set up, with the additional ease that comes with a cable-free environment.

Backward Compatibility

Wireless USB is backward compatible to wired USB devices.

9.1.7 Ultra-Wideband (UWB)**9.1.7.1 Background**

Ultra-Wideband (UWB) is a technology for transmitting information spread over a large bandwidth that should, in theory and under the right circumstances, be able to share spectrum with other users.

Ultra Wideband was traditionally accepted as impulse radio, but the FCC and ITU-R now define UWB in terms of a transmission from an antenna for which the emitted signal bandwidth exceeds the

lesser of 500 MHz or 20% bandwidth. Thus, pulse-based systems – where in each transmitted pulse instantaneously occupies a UWB bandwidth, or an aggregation of at least 500 MHz worth of narrow band carriers, for example in orthogonal frequency-division multiplexing (OFDM) fashion – can gain access to the UWB spectrum under the rules. Pulse repetition rates may be either low or very high. Pulse-based radars and imaging systems tend to use low repetition rates, typically in the range of 1 to 10 megapulses per second.

On the other hand, communications systems favour high repetition rates, typically in the range of 1 to 2 gigapulses per second, thus enabling short-range gigabit-per-second communications systems. Each pulse in a pulse-based UWB system occupies the entire UWB bandwidth, thus reaping the benefits of relative immunity to multi path fading (but not to inter symbol interference), unlike carrier-based systems that are subject to both deep fades and inter symbol interference.

The FCC power spectral density emission limit is the same as for unintentional emitters in the UWB band, but is significantly lower in certain segments of the spectrum.

A significant difference between traditional radio transmissions and UWB radio transmissions is that traditional transmissions transmit information by varying the power/frequency/and or phase in distinct and controlled frequencies while UWB transmissions transmit information by generating radio energy at specific times with a broad frequency range [26].

One of the valuable aspects of UWB radio technology is the ability for a UWB radio system to determine "Time of Flight" of the direct path of the radio transmission between the transmitter and receiver. With any radio transmission the signals reflect off of metallic objects and result in different radio signal paths that then can arrive at the receiver later in time and interfere with radio transmission that went directly from the transmitter to the receiver. With frequency based transmissions the sinusoidal waves add/subtract at the receiver antenna and make it difficult or impossible to distinguish the direct transmission path from the reflected paths. This is called "multi-path fading" and "multi-path interference". However, with UWB transmissions the time encoding can be randomly dithered and the receiver can then determine which is the direct path. With a bidirectional system or a radar system this allows distances to be determined much more accurately.

9.1.7.2 Advantages over narrow band

Channel capacity

A UWB link will have a far greater channel capacity (maximum data rate) than will a narrow band link that uses the same transmit power. Said another way, the UWB link will go much farther than the narrow band link, using the same transmit power and same data rate. This same efficiency improvement applies to other uses of radio such as position, location, tracking, and radar. Typical improvements can be several orders of magnitude.

Precise distance measurement

When using radio signals to measure distances, one must place some abrupt change in the signal at some point in time so that a precise timing measurement can be made. Nature requires that such an abrupt change in a signal contain wideband energy. A channel whose bandwidth is narrow cannot convey any abrupt changes in the signal, by definition. Thus UWB can be used for precise distance measurement, position, location, tracking, and radar.

Stealth

When stealth is required, it is easy to design a UWB signal that looks like nothing more than background noise to a receiver that is unaware of the signal's coding. Also, the fact that far less power is required to accomplish any task makes detection proportionally more difficult.

High packing rate

More channels can be packed into a given band. This is because each link requires so much less power. Note that this is not so much the case if the UWB links are designed without proper coding and error correction. This is because there is always some slight cross talk between UWB channels, and a purely analogue UWB channel, for example, would pick this up, while a digital channel with proper coding and error correction could completely eliminate the cross talk. In practice, however, this is easier said than done.

9.2 Services Discovery

9.2.1 Bluetooth

Bluetooth is a radio standard and communications protocol primarily designed for low power consumption, with a short range (power class dependent: 1 meter, 10 meters, 100 meters) based around low-cost transceiver microchips in each device.

Bluetooth lets these devices communicate with each other when they are in range. The devices use a radio communications system, so they do not have to be in line of sight of each other, and can even be in other rooms, so long as the received power is high enough. As a result of different antenna designs, transmission path attenuations, and other variables, observed ranges are variable; however, transmission power levels must fall into one of three classes:

Class	Maximum Permitted Power (mW)	Maximum Permitted Power (dBm)	Range (approximate)
Class 1	100 mW	20 dBm	~100 meters
Class 2	2.5 mW	4 dBm	~10 meters
Class 3	1 mW	0 dBm	~1 meter

The Bluetooth specification was first developed by Ericsson (now Sony Ericsson and Ericsson Mobile Platforms), and was later formalized by the Bluetooth Special Interest Group (SIG). Bluetooth is also known as IEEE 802.15.1.

9.2.1.1 Bluetooth 1.0 and 1.0B

Versions 1.0 and 1.0 B had numerous problems and the various manufacturers had great difficulties in making their products interoperable. 1.0 and 1.0B also had mandatory Bluetooth Hardware Device Address (BD_ADDR) transmission in the handshaking process, rendering anonymity impossible at a protocol level, which was a major setback for services planned to be used in Bluetooth environments, such as Consumerium.

9.2.1.2 Bluetooth 1.1

- Many errors found in the 1.0B specifications were fixed.
- Added support for non-encrypted channels.
- Received Signal Strength Indicator (RSSI)

9.2.1.3 Bluetooth 1.2

This version is backwards compatible with 1.1 and the major enhancements include

- *Adaptive Frequency-hopping spread spectrum (AFH)*, which improves resistance to radio frequency interference by avoiding the use of crowded frequencies in the hopping sequence
- *Higher transmission speeds* in practice
- *Extended Synchronous Connections (eSCO)*, which improves voice quality of audio links by allowing retransmissions of corrupted packets.
- *Host Controller Interface (HCI) support for 3-wire UART*
- *HCI access to timing information* for Bluetooth applications:

9.2.1.4 Bluetooth 2.0

This version is backwards compatible with 1.x. The main enhancement is the introduction of *Enhanced Data Rate (EDR)* of 3.0 MBps. This has the following effects (Bluetooth SIG, 2004):

- 3 times faster transmission speed (up to 10 times in certain cases).
- Lower power consumption through a reduced duty cycle.
- Simplification of multi-link scenarios due to more available bandwidth.
- Further improved BER (bit error rate) performance.

9.2.1.5 The future of Bluetooth

The next version of Bluetooth, currently code named Lisbon, includes a number of features to increase security, usability and value of Bluetooth. The following features are defined:

- Atomic Encryption Change – allows encrypted links to change their encryption keys periodically, increasing security, and also allowing role switches on an encrypted link.
- Extended Inquiry Response – provides more information during the inquiry procedure to allow better filtering of devices before connection. This information includes the name of the device, and a list of services, with other information.
- Sniff Subrating – reducing the power consumption when devices are in the sniff low power mode, especially on links with asymmetric data flows. Human interface devices (HID) are expected to benefit the most with mice and keyboards increasing the battery life from 3 to 10 times those currently used.
- QoS Improvements – these will enable audio and video data to be transmitted at a higher quality, especially when best effort traffic is being transmitted in the same piconet.
- Simple Pairing – this improvement will radically improve the pairing experience for Bluetooth devices, while at the same time increasing the use and strength of security. It is expected that this feature will significantly increase the use of Bluetooth.

Bluetooth technology already plays a part in the rising Voice over IP (VOIP) scene, with Bluetooth headsets being used as wireless extensions to the PC audio system. As VOIP becomes more popular, and more suitable for general home or office users than wired phone lines, Bluetooth may be used in Cordless handsets, with a base station connected to the Internet link.

9.2.1.6 Communication, connection

A Bluetooth device playing the role of the "master" can communicate with up to 7 devices playing the role of the "slave". This network of "group of up to 8 devices" (1 master + 7 slaves) is called a piconet. A piconet is an ad-hoc computer network of devices using Bluetooth technology protocols to allow one master device to interconnect with up to seven active slave devices (because a three-bit MAC address is used). Up to 255 further slave devices can be inactive, or parked, which the master device can bring into active status at any time.

At any given time, data can be transferred between the master and 1 slave; but the master switches rapidly from slave to slave in a round-robin fashion. (Simultaneous transmission from the master to multiple slaves is possible, but not used much in practice). Either device may switch the master/slave role at any time.

9.2.2 Jini

The Jini architecture specifies a way for clients and services to find each other on the network and to work together to get a task accomplished. Service providers supply clients with portable Java technology-based objects ("Java objects") that give the client access to the service. This network interaction can use any type of networking technology such as RMI, CORBA, or SOAP, because the

client only sees the Java object provided by the service and, subsequently, all network communication is confined to that Java object and the service whence it came.

When a service joins a network of Jini technology-enabled services or devices, it advertises itself by publishing a Java object that implements the service API. This object's implementation can work in any way the service chooses. The client finds services by looking for an object that supports the API. When it gets the service's published object, it will download any code it needs in order to talk to the service, thereby learning how to talk to the particular service implementation via the API. The programmer who implements the service chooses how to translate an API request into bits on the wire using RMI, CORBA, XML, or a private protocol [7].

The existence of the Java platform makes it possible to define the Jini networking technology which, in turn, enhances the value of the Java platform by making services available throughout the network. The Java platform specifies what is available on any particular machine that is running the platform. It defines a set of services (classes and the Java Virtual Machine) that exist on a particular machine that can be used by the programs running on that machine [8]. The Jini technology extends this notion of a platform from a particular machine to the network that connects machines which are running the Java platform. Jini technology-enabled services are not necessarily resident on any particular machine in the network, but are instead available to all of the machines through the network. Services do not need to be everywhere, but instead only need to be somewhere on the network to be available to all of the participants in the network.

9.2.3 SLP

The Service Location Protocol (SLP) provides a scalable framework for the discovery and selection of network services. Using this protocol, computers using the Internet no longer need so much static configuration for network services for network-based applications. This is especially important as computers become more portable and users less tolerant or able to fulfil the demands of network system administration.

Traditionally, users find services by using the name of a network host (a human readable text string), which is an alias for a network address. SLP eliminates the need for a user to know the name of a network host supporting a service. Rather, the user names the service and supplies a set of attributes, which describe the service. SLP allows the user to bind this description to the network address of the service.

SLP provides a dynamic configuration mechanism for applications in local area networks. It is not a global resolution system for the entire Internet; rather it is intended to serve enterprise networks with shared services. Applications are modelled as clients that need to find servers attached to the enterprise network at a possibly distant location. For cases where there are many different clients or services available, the protocol is adapted to make use of nearby Directory Agents that offer a centralized repository for advertised services. The basic operation in SLP is that a client attempts to discover the location for a service. In small installations, each service is configured to respond individually to each client. In larger installations, service will register their services with one or more directory agents and clients contact the directory agent to fulfil request for service location information. This is intended to be similar to URL specifications and makes use of URL technology.

9.2.4 Universal PnP

The UPnP architecture offers pervasive peer-to-peer network connectivity of PCs, intelligent appliances, and wireless devices [9]. The UPnP architecture is a distributed, open networking architecture that uses TCP/IP and HTTP to enable seamless proximity networking in addition to control and data transfer among networked devices in the home, office, and everywhere in between.

It enables data communication between any two devices under the command of any control device on the network.

- Media and device independence. UPnP technology can run on any medium including phone lines, power lines (PLC), Ethernet, IR (IrDA), RF (Wi-Fi, bluetooth), and FireWire. No device drivers are used; common protocols are used instead.

- Common base protocols. Base protocol sets are used, on a per-device basis.
- User interface (UI) Control. UPnP architecture enables vendor control over device user interface and interaction using the web browser.
- Operating system and programming language independence. Any operating system and any programming language can be used to build UPnP products. UPnP does not specify or constrain the design of an API for applications running on control points; OS vendors may create APIs that suit their customer's needs. UPnP enables vendor control over device UI and interaction using the browser as well as conventional application programmatic control.
- Internet-based technologies. UPnP technology is built upon IP, TCP, UDP, HTTP, and XML, among others.
- Programmatic control. UPnP architecture also enables conventional application programmatic control.
- Extendable. Each UPnP product can have value-added services layered on top of the basic device architecture by the individual manufacturers.

The UPnP architecture supports zero-configuration, *invisible* networking and automatic discovery for a breadth of device categories from a wide range of vendors, whereby a device can dynamically join a network, obtain an IP address, announce its name, convey its capabilities upon request, and learn about the presence and capabilities of other devices. DHCP and DNS servers are optional and are only used if they are available on the network. A device can leave a network smoothly and automatically without leaving any unwanted state information behind.

The foundation for UPnP networking is IP addressing. Each device must have a Dynamic Host Configuration Protocol (DHCP) client and search for a DHCP server when the device is first connected to the network. If no DHCP server is available, that is, the network is unmanaged, the device must assign itself an address. If during the DHCP transaction, the device obtains a domain name, for example, through a DNS server or via DNS forwarding, the device should use that name in subsequent network operations; otherwise, the device should use its IP address.

9.2.4.1 Protocol

Discovery

Given an IP address, the first step in UPnP networking is discovery. When a device is added to the network, the UPnP discovery protocol allows that device to advertise its services to control points on the network. Similarly, when a control point is added to the network, the UPnP discovery protocol allows that control point to search for devices of interest on the network. The fundamental exchange in both cases is a discovery message containing a few, essential specifics about the device or one of its services, for example, its type, identifier, and a pointer to more detailed information. The UPnP discovery protocol is based on the Simple Service Discovery Protocol (SSDP).

Description

The next step in UPnP networking is description. After a control point has discovered a device, the control point still knows very little about the device. For the control point to learn more about the device and its capabilities, or to interact with the device, the control point must retrieve the device's description from the URL provided by the device in the discovery message. The UPnP description for a device is expressed in XML and includes vendor-specific, manufacturer information like the model name and number, serial number, manufacturer name, URLs to vendor-specific web sites, etc. The description also includes a list of any embedded devices or services, as well as URLs for control, eventing, and presentation. For each service, the description includes a list of the commands, or actions, to which the service responds, and parameters, or arguments, for each action; the description for a service also includes a list of variables; these variables model the state of the service at run time, and are described in terms of their data type, range, and event characteristics.

Control

The next step in UPnP networking is control. After a control point has retrieved a description of the device, the control point can send actions to a device's service. To do this, a control point sends a

suitable control message to the control URL for the service (provided in the device description). Control messages are also expressed in XML using the Simple Object Access Protocol (SOAP). Like function calls, in response to the control message, the service returns any action-specific values. The effects of the action, if any, are modelled by changes in the variables that describe the run-time state of the service.

Event notification

The next step in UPnP networking is event notification, or "eventing". A UPnP description for a service includes a list of actions the service responds to and a list of variables that model the state of the service at run time. The service publishes updates when these variables change, and a control point may subscribe to receive this information. The service publishes updates by sending event messages. Event messages contain the names of one or more state variables and the current value of those variables. These messages are also expressed in XML and formatted using the General Event Notification Architecture (GENA). A special initial event message is sent when a control point first subscribes; this event message contains the names and values for all evented variables and allows the subscriber to initialize its model of the state of the service. To support scenarios with multiple control points, eventing is designed to keep all control points equally informed about the effects of any action. Therefore, all subscribers are sent all event messages, subscribers receive event messages for all "evented" variables that have changed, and event messages are sent no matter why the state variable changed (either in response to a requested action or because the state the service is modelling changed).

Presentation

The final step in UPnP networking is presentation. If a device has a URL for presentation, then the control point can retrieve a page from this URL, load the page into a web browser, and depending on the capabilities of the page, allow a user to control the device or view device status. The degree to which each of these can be accomplished depends on the specific capabilities of the presentation page and device.

9.2.5 HAVi

HAVi is a digital AV networking initiative that provides a home networking software specification for seamless interoperability among home entertainment products [11]. Equally important, the HAVi specification is AV-device-centric, so it has been designed to meet the particular demands of digital audio and video. It defines an operating-system-neutral middleware that manages multi-directional AV streams, event schedules, and registries, while providing APIs for the creation of a new generation of software applications. Whatever their brand is, the focus is on the control and content of digital AV streams. HAVi software takes advantage of the powerful resources of chips built into modern audio and video appliances to give you the management function of a dedicated audio-video networking system.

The HAVi Specification was developed for home entertainment AV networks, providing high bandwidth for transmitting multiple AV streams and featuring easy "plug-and-enjoy" functionality, using an underlying IEEE-1394 digital interface (i.LINK™ or FireWire™).

The HAVi specification defines a set of APIs and middleware capable of automatically detecting devices on the network, coordinating the functions of various devices, installing applications and user interface software on each appliance, and ensuring interoperability among multiple brands of devices [12].

The HAVi Organization promotes the adoption of the HAVi architecture and the development of interconnecting "bridges" with other home networking standards such as Jini and Universal Plug and Play (UPnP).

The HAVi Specification is primarily designed as a distributed system for providing interoperability and plug and play capabilities of Audio and Video Systems. It is optimized for transfer and management of high bandwidth digital AV streams with very high Quality of Service requirements.

This protocol may not be optimal for controlling home appliance such as washers, dryers and for light switches.

Interoperability:

Functions on a device within the HAVi networking system may be controlled from another device within the system. Search for an available VCR to record a TV program, with commands being given via the menu selection of another TV display.

Brand independence:

Entertainment products from different manufacturers will communicate with each other when connected into a HAVi network. Imagine a variety of VCR's, hi-fis, DVD players, MiniDisc machines, active loudspeakers, set-top boxes all daisy-chained together and showing up on the TV for you to control from your one remote commander!

Hot "Plug and Enjoy:"

HAVi compliant devices automatically announce their presence and capabilities to every other device on the HAVi network, greatly simplifying installation and setup. Just plug-and-enjoy. No more complicated and difficult installation instructions. No configuration of network addresses or device drivers.

Linked to the Past, Upgradeable in the future:

Today's i.LINK enabled camcorders and other devices will be able to be controlled on a HAVi network for basic functions. And most HAVi compliant devices will come with their own dynamic Device Control Modules. Updating functionality can be done by downloading/uploading new capabilities via the Internet. Also, additional or replacement products can simply be incorporated into the network.

9.2.6 Salutation

9.2.6.1 Background

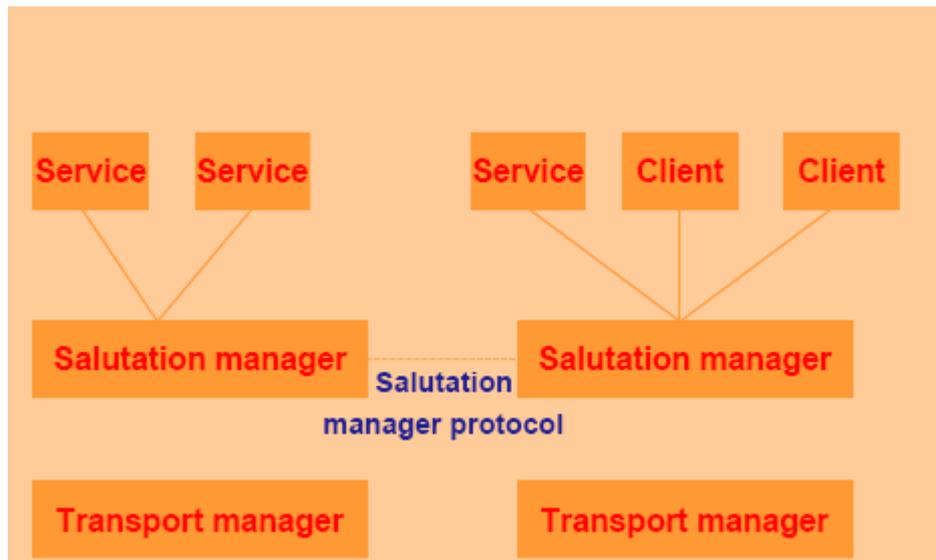
The open Salutation Architecture provides a standard method for applications, services and devices to advertise their capabilities to their counterparts over a network, and search for and use the capabilities of

other applications, services and devices when needed. Because a large variety of appliances and equipment may be interconnected, the "network broker" architecture is independent of processors, operating systems and communications protocols. It also allows for scalable implementations, even in very low-price devices. The Salutation Bluetooth Profile has been developed by the Consortium with contributions from IBM [27].

9.2.6.2 Architecture

It contains three main components:

- Services that can register and unregister functional units with the local Salutation manager
- Salutation managers that function as service brokers
- Transport managers from the details of specific network transport protocols



A client can use the searchCapability call to determine if Salutation managers have registered specific functional units.

9.2.6.3 Salutation manager

A Salutation manager can operate in one of three "personalities":

- In native personality, Salutation managers are used only for discovery.
- The emulated personality is similar to the native personality in that Salutation managers set up the connection, but in this case they transfer native data packets encapsulated in Salutation manager protocol format (bridge)
- In Salutation personality, Salutation managers establish the connection between client and service, and they also mandate the specific format of the data transferred. The Salutation architecture defines the data formats.

9.2.6.4 Transport manager

Transport managers also locate the Salutation managers on their respective network segments via either multicast, static configuration, or reference to a centralized directory.

- Discovery of other Salutation managers allows a particular Salutation manager to determine which functional units have been registered and to allow clients access to these remote services.
- Communication between Salutation managers is based on remote procedure call (RPC).

9.2.7 Comparison between different service discovery systems

	JINI	Salutation	SLP	UPnP	Bluetooth
Main entities	Lookup Service, Client, Service	Salutation Manager, Transport Manager, Client Server	Directory Agent, Service Agent, User Agent	Control Point, Devices (Services)	SDP Client, SDP Server (or both)
Service repository	Lookup Service	A set of SLMs	DA (directory agent)	None	SDP Server

	JINI	Salutation	SLP	UPnP	Bluetooth
Service announcement	Discovery/Join protocol	Registering with local SLM	Service Registration	Multicast advertisement	Not Supported
Access to service	Service proxy object based on RMI	Service Session Management	Service type for discovered service	Invoking Action to service	Not Supported
Service description	Interface type and attribute matching	Functional Unit and attributes within it	Service type and attribute matching	Description in XML	Attribute ID and Attribute Value
Service group	Group	No	Scope	No	Service Class
Event notification	Remote Events	Availability Checking (periodic & automatic)	SLP extension for event notification	Service publishes event when state variable changes	Not Supported
Other features	Java-centric architecture	Transport independence	Authentication security feature	Automatic configuration	Services could be browsed from a hierarchy
Usage	CNN and sprint Web/Directory Servers, E-mail, Calendar, Collaboration Servers		Novell Netware	WinXP for gateways, Internet connectivity and NAT	Bluetooth access points, print adaptors, Palm OS bluetooth system

9.3 Wireless Devices

Wireless devices are important for context sensing as an important issue constraining the definition of the Hydra middleware. For positioning and location detection as an important part of sensed contextual information there are in principal three techniques which can be distinguished:

1. *Triangulation* the geometric properties of triangles to calculate the location of the target. This technique is further divisible into the two subcategories of *lateration* (e.g., Time delay on arrival (TDOA) [13]) which makes use of distance to target, and *angulation* which makes use of the angle or bearing to target (e.g., VHF Omni directional Ranging (VOR) aircraft navigation system).
2. *Scene analysis* uses observed information gathered about an environment from a known reference point to draw conclusions about that environment or its member targets. Such techniques include image analysis or RSSI measurements of point targets (e.g., Ekahau [14] or RADAR [15] positioning system).
3. *Proximity analysis* refers to any technology which detects the presence of a target within a fixed distance of a sensor. The plethora of technologies includes RFID [16], pressure sensitivity, infrared and capacitive sensing. (Also: active badge [17])

All three types can be used for tracking objects and users either *indoor* or *outdoor*.

Each of them has advantages and disadvantages in comparison to the others in terms of the various characteristics for evaluating the according system. This characteristics include the *accuracy* (maximum deviation of provided data from true position, e.g. 5m for GPS), the *granularity* (e.g., scalar value or room name) and *recognition rate* which is the ability of the system to assign tracking data to a tracked asset (this ability might be decreased in scene analysis due to problems w.r.t. identifying users).

There is also another (more conceptual) classification [19] into two major categories.

- The first category is *receptive* localization. The positioning data is effectively *broadcasted* within a certain range and the mobile device can derive its own location from this data. GPS is an example of this. For acquiring the actual positioning the mobile device has to infer the actual location (symbolic, coordinates) from this data using a map.
- The second category is *transmissive* localization. The position is computed by a fixed station which either perceives the mobile device or receives the tracking data from a beacon which perceives it. The station can then either transmit the derived location information back to the mobile device, or use it internally to for instance grant access to a service for the mobile user. Positioning within cellular networks (e.g., triangulation in 2/3G networks) is an example this, where the beacon of the mobile communication channel is also used for positioning.

The according technologies are reflected in different sorts of wireless devices.

9.3.1 RFID tags

Radio Frequency IDentification (RFID) is an automatic identification method using radio waves. It relies on storing and remotely retrieving data using devices called *RFID tags* or *transponders*. There is a long history of the RFID technology which can be directly related to a conceptually equivalent techniques employed by the allied forces in World War II called IFF (Identification Friend or Foe).

An RFID tag is a small scale wireless device comprising one or more silicon (alternatively: polymer) chips and antennas that can be attached to or incorporated into an object (product, asset, animal, or person) for the purpose of identification. Chip-based RFID tags contain silicon chips and antennas. Passive tags require no internal power source, whereas active tags require a power source. Moreover, semi-passive / semi-active variants exist.

- *Passive tags*: use the current induced in the antenna by the incoming radio frequency signal (e.g., asking for the identification of the tag) to do data processing and transmission. Such devices can be fairly small: the smallest such devices measured 0.15 mm × 0.15 mm, and are thinner than a sheet of paper (7.5 micrometers) [18]. Commercially available products exist that can be embedded under the skin. Passive tags have practical read distances ranging from about 10 cm (ISO 14443) up to a few meters (ISO 18000-6) which depends on the chosen radio frequency and antenna design/size. Non-silicon tags made from polymer semiconductors are currently being developed by several companies. Simple laboratory printed polymer tags operating at 13.56 MHz were demonstrated in 2005 by both PolyIC (Germany) and Philips (The Netherlands). The advantage of this production technique is mainly the inexpensiveness when compared to silicon chips. Thus polymer RFID tags are a candidate to overcome the conventional barcode.
- *Active tags*: such tags have their own internal power source which results in a more reliable communication between the tag and the outside environment. At present, the smallest active tags are about the size of a coin and sell for a few dollars. Usually, they contain sensors, e.g. for sensing and logging of temperatures to monitor the temperature of perishable goods or other physical measure (humidity, air pressure etc). The battery life can be up to 10 years and they can typically communicate over ranges of hundreds of meters.

The four most common tags in use are categorized by radio frequency:

- low frequency tags (125 or 134.2 kHz),
- high frequency tags (13.56 MHz),
- UHF tags (868 to 956 MHz), and
- Microwave tags (2.45 GHz).

Some standards that have been made regarding RFID technology include:

- ISO 11784 & 11785: these standards regulate the Radio frequency identification of animals in regards to Code Structure and Technical concept

- ISO 14223/1: radio frequency identification of animals, advanced transponders - Air interface
- ISO 10536: close coupled cards
- ISO 14443: proximity cards
- ISO 15693: vicinity cards
- ISO 18000: RFID for item management; air interface
- EPCglobal: this is the standardization framework includes major companies around the world such as Wall Mart and the Gillette Company. The objective is to eventually use the EPC (Electronic Product Code) and RFID to identify any item, in any industry, anywhere in the world.

Though the technology is quite promising and is expected to change the future world, significant security concerns exist. Yet, a secure form of RFID for the consumer population is lacking in which privacy issues are resolved.

9.3.2 Active Badges

Location tracking of assets through an active badge system demands for featuring a set of wireless and wired devices. The Badge System can locate a user inside/at a special location (e.g. room, PC ...) and principal idea is quite similar to RFID. The main physical devices are the *badge* which presents the user or the object which is going to be tracked, the *beacon* (infrared transmitter) representing the symbolic location and the *base station* which handles the incoming requests from the system. The badge as an (active) tag (therefore referred to as an 'Active Badge') emits a unique code within a certain period of time, e.g. for approximately a tenth of a second every 15 seconds. The emitted periodic signals are picked up by a network of mobile sensors moving around a site or campus. A master station, also connected to the network, polls the sensors for badge 'sightings', processes the received data, and then makes it available to clients being attached to the network for making use of the context data.

In the first of deployment of an Active Badge system at Olivetti Research (ORL) and Cambridge University (UK) [17] it was used to track staff members wearing badges (the badge was designed in a package roughly 55x55x7mm and had a weight of 40g which was considered to be comfortable) that transmit signals providing information about their location to a centralized location service. The location information then was used for location-dependent telephone call routing. Later the location tracking was complemented by weight-measuring (*Active Floor*) and the (transmissive) ORL ultrasonic location system for relative positioning of users and devices.

In a more recent implementation [20], pulse-width modulated infrared (IR) signals were used for signalling between the badge and sensor. This was because IR solid-state emitters and detectors can be made very small and very cheaply (in contrast to ultrasonic transducers); they can be made to operate with a 6m range, and the signals are reflected by partitions and therefore are not directional when used inside a small room. Moreover, the signals cannot travel through walls, unlike radio signals that can penetrate the partitions founding office buildings.

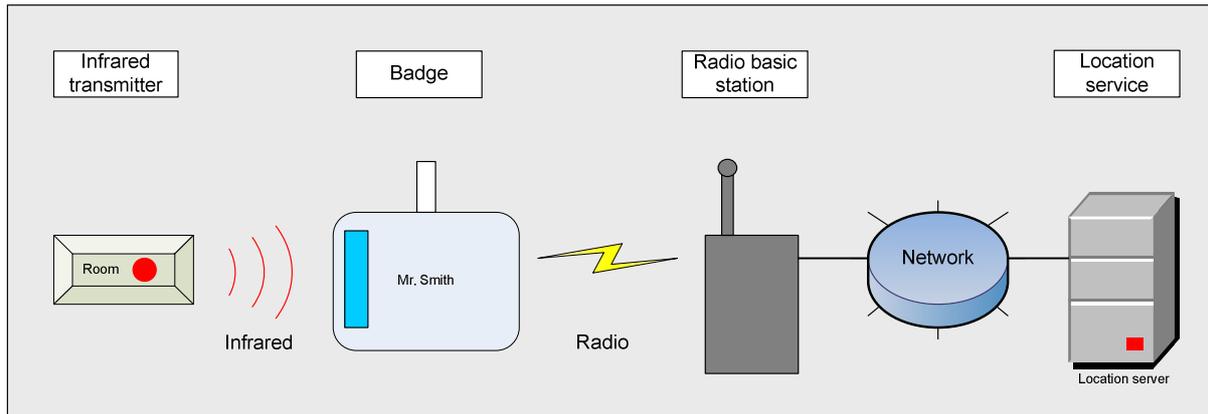


Figure 10 Badge system

In this setup, the following components are featured (see Figure 10):

- *Stationary, stand-alone infrared ID beacons:* the beacons are not-networked tags scattered over the site where users / devices are tracked. They periodically emit location information (the location ID) in programmable time cycles. A PIC reads a "silicon serial number" and generates the ID bursts. The emitted signal strength (up to 15 mW) can easily be modified thereby adjusting the maximum transmission distance.
- *Mobile, IR receiving and RF transmitting badges:* the badges are sized similar to a corporate ID card, slightly thicker than a credit card. They provide sufficient space for the photograph and name of the wearer. The badge, just, listens to IR location IDs in varying time cycles, depending on the motion of the badge (short cycles while moving, longer cycles while the badge is stopped, immediate scanning when a button is pressed).
- *Stationary RF receivers (radio base station) with LAN (or WLAN) interface:* the RF base station has been designed as simple as possible. It incorporates a low cost micro controller module (Dallas Semiconductor TINI) that offers a Real Time Operating System (RTOS) and a built-in Java Virtual Machine (JVM). Via the Ethernet interface, the software of the base station allows to query the location information.

9.3.3 GPS-enabled devices

The Global Positioning System (NAVSTAR GPS or GPS for short) yet is the only fully-functional global satellite navigation system used for outdoor positioning. It is partly based on a similar ground-based radio navigation systems called LORAN (Long-range Radio Aid to Navigation), developed in the early 1940s. The European Union is currently developing and establishing *Galileo* as an alternative to the US owned GPS system which is planned to be operational by 2010. China, Israel, India, Morocco, Saudi Arabia, and South Korea in the meantime have joined the EU initiative.

The system makes use of a constellation of more than 24 GPS satellites which broadcast precise timing signals by radio, allowing any GPS receiver to accurately determine its location (longitude, latitude, and altitude) independent weather, daytime and location conditions. Thus, GPS is a receptive positioning system which can be used in latitude of application domains.

For instance, in agriculture machine guidance of tractors and other large agricultural machines is done via GPS. In such scenarios, location information is needed to semi-automatically steer and visualize contextual information on procedures and working tasks such as row crop operations and when spraying. Additional to guidance, GPS is used in harvesters with yield monitors where it can provide a yield map of the paddock being harvested. GPS functionality can be also used by emergency services and location-based services to locate mobile phones, or in location based gaming (geo caching) in which a user has to travel to a specific longitude and latitude to search for objects concealed by other players.

GPS enabled devices come in a variety of form factors, ranging from devices integrated into cars, phones, and watches, to dedicated devices. They usually integrate several radio receivers [21] which are needed to tune to several satellites simultaneously for computing the accurate positioning based on the data broadcasted by the GPS-satellites:

1. The first type of data of the so called *almanac* which contains coarse time information with second precision along with status information about the satellites.
2. The second is the *ephemeris*, which contains orbital information of the particular satellite that allows the receiver to calculate the position of the satellite at any point in time.
3. The third consists of two forms of *accurate clock* information, the *coarse acquisition code*, or C/A, and the *precise code*, or P-code. The C/A code resulting in reduced resolution is normally used for most civilian navigation whereas the latter is used in military applications.

The whole data is mixed together and broadcasted via the primary radio channel L1 (1575.42 MHz). Based on this data a radio receiver can calculate the distance from GPS satellite sending this data. Merging the distances computed by a set of different radio receivers, the absolute coordinates (longitude, latitude, and altitude, thus yielding the location) of the device the receivers are attached to can be gained.

The number of such radio receivers gives an indication for the accuracy of the location data to be expected and the expense for a GPS device critically depends on it. Early consumer-level receivers typically included six to eight receivers. Due to the reduced cost of implementing the receiver chips, today even low-cost hand held receivers typically comprise twelve receivers. Currently, the SiRF Star III chip set (20 channels) is quite popular and used in high-quality products. Even more expensive units, known as "dual-frequency receivers", additionally to L1 also tune in the L2 radio channel (1227.60 MHz) signals in order to compensate ionospheric delays thus yielding enhanced accuracy.

GPS receivers are usually bundled with other components to form a GPS-enabled device. Yet, the following GPS popular device classes exist:

- *GPS mouse*, i.e., receiver
 - either with *wireless* interface (e.g., Bluetooth). Examples: Haicom HI-406 BT, Leadtek BT 9553X, Wintec WBT-200, TomTom Wireless GPS, Cellink BTG-7000, SysOnChip Bluetooth SMART GPS.
 - or with *wired* computer interface (e.g., RS.232, USB, PCMCIA, SD or Compact Flash interface). Examples: HOLUX GR-271, Wintec WGM-300 USB, Rikaline 6010, Naviflash 1060. Nokia offers a shell as wrapper for Nokia 5140, 5140i CC-70D phones called Xpress-on GPS.

Such receivers are intended to be combined with another (wireless or wired) device (e.g., mobile / cell phone) for providing access to further services and functions (maps, POI information etc.) and serving the communication back-link (thus enabling *transmissive* positioning functions).

- GPS receiver as part of an
 - *all-in-one navigation system* for locating and routing vehicles based on maps and suggesting POIs (points of interest). Such systems frequently integrate speech output in order to facilitate convenient use. Examples: Garmin StreetPilot, Garmin GPS MAP 60Cx, TRANSONIC MOBILE NAVIGATOR 1.0 GPS, Umax VAOVA Travel-100, Yakumo PNA EazyGo.
 - or in a *mobile computer* (PDA or laptop). Alternatively to a single-purpose navigation system, GPS receivers have been integrated into PDAs or notebooks. Examples: Fujitsu Siemens Computers Pocket Loox T Series / Loox N Series, Acer n35 Handheld PDA, Palm Zire 31 GPS, Hewlett-Packard iPaq 2110, Asus MyPal A632.

Currently, this is the most popular form factor of receiver devices for the civilian use of GPS.

Through the FCC Enhanced 911 (E911) mandate it is intended to dictate location tracking capabilities inside all new cell phone handsets. Although the original deadline of Oct. 1, 2001 was

missed, the advantages presented by GPS enabled cell phones in rescue situations are appealing. Moreover, the room for improvements of GPS is currently explored. Newer and enhanced forms of GPS, such as *differential GPS* and *carrier-phase GPS*, can enable receivers to give theoretically up to 3 or 4mm accuracy.

9.4 References

- [1] Bluetooth SIG. Bluetooth Special Interest Group Launches Bluetooth Core Specification Version 2.0 + Enhanced Data Rate. Press release.
- [2] ZigBee Alliance, "ZigBee Specifications", version 1.0, April 2005.
- [3] F. Zhao, L. Guibas, *Wireless Sensor Networks – An Information Processing Approach*, Morgan Kaufman Publisher, S. Francisco 2004.
- [4] John P. Walthers, Zhengqiang Liang, Weisong Shi, and Vipin Chaudhary, "Wireless Sensor Networks Security: A Survey", Technical Report MIST-TR-2005-007, July 2005.
- [5] T. Rappaport, "Wireless Communications: Principles and Practice, 2nd Ed. Prentice Hall, Upper Saddle River, NJ, 2002.
- [6] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*, Wiley, 2005.
- [7] Core JINI, Edwards and Joy, Prentice Hall, June 28 1999.
- [8] The Jini Specification, Arnold, Wollrath, O'Sullivan, Scheifler, Waldo, Addison-Wesley, June 1999.
- [9] UPnP Forum, UPnP Home Page, <http://www.upnp.org>
- [10] Bluetooth SIG. Bluetooth Special Interest Group Launches Bluetooth Core Specification Version 2.0 + Enhanced Data Rate. Press release.
- [11] K.-W. Hsu, C.-L.. Chen, W.-C. Li, and T.-Y. Yu, A Message Delivery Mechanism for HAVi Network, 2006
- [12] HAVi, <http://www.havi.org>
- [13] D. Cook, S. Das. *Smart Environments: Technology, Protocols and Applications*, Wiley Series on Parallel and Distributed Computing, 2004.
- [14] The Ekahau Positioning System Homepage, www.ekahau.com, Oct/10/06.
- [15] J. R. Hightower, G. Borriello. Location systems for ubiquitous computing. *Computer*, 34(8): 57-66, August 2001.
- [16] R.J. Orr, G.D. Abowd, D. Salber. The Smart Floor: A Mechanism for Natural User Identification and Tracking, - Proc. 2000 Conf. Human Factors in Computing Systems.
- [17] R. Want, A. Hopper, V. Falcao, J. Gibbons. The active badge location system, *ACM Transactions on Information Systems*, January 1992.
- [18] Yoshiko Hara: Hitachi advances paper-thin RFID chip, *EE Times*, (02/06/2006 7:00 AM EST), <http://www.eetimes.com/news/design/showArticle.jhtml?articleID=179100286>.
- [19] Long, S.; Kooper, R.; Abdowd, G.D.; Atkeson, C.G: Rapid Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study. - in Proc. 2nd ACM Intl. Conf. On Mobile Computing and Networking, MobiCom 1996.
- [20] Tom Pfeifer, Dirk Elias: Commercial Hybrid IR/RF Local Positioning System, KiVS (Kommunikation in Verteilten Systemen), Feb 26-28 2003, University of Leipzig, Germany.
- [21] James Bao-Yen Tsui: *Fundamentals of Global Positioning System Receivers. A Software Approach*. John Wiley & Sons Inc; 2nd edition (January 2005).
- [22] Wi-Fi, <http://www.wi-fi.org/>
- [23] HomeRF white papers, <http://www.cazitech.com/HomeRF%20WP%20%20Home%20NW%20Technologies.PDF>

- [24] Z-Wave, <http://www.zen-sys.com/index.php?page=32>
- [25] Wireless USB, <http://www.usb.org/developers/wusb/>
- [26] UWB, <http://www.intel.com/technology/comms/uwb/>
- [27] Salutation Consortium, <http://www.palowireless.com/infotooth/knowledge/othernetworks/126.asp>

10 Privacy and security

10.1 Privacy at the middleware and application layers

10.1.1 Privacy solutions at the middleware/application layers

Privacy is a state or condition of limited access to a person [1]. Information privacy is usually concerned with the confidentiality of personal identifiable information (PII) and with the individual's right to determine how, when, and to what extent information about oneself will be released to another person or an organization [2]. Technical mechanisms that are used to protect privacy are divided into four broad categories, encryption and security mechanisms, privacy enhancement technologies that include a variety of anonymizing and de-identifying techniques, infrastructures - privacy aware software and labelling protocols [3].

10.1.1.1 Access control

Access control is the process of limiting access to the resources of a system only to authorized users, programs, processes, or other systems. In general, it is defined as the mechanism by which users are permitted access to resources according to their identities authentication and associated privileges authorization [4].

Discretionary Access Control

Discretionary Access Control (DAC) is a means of restricting access to objects based on the identity and need-to-know of users or membership in certain groups [5, 6, 7]. It is an access policy determined by the owner of a resource who controls access to the object. The concept is utilized at most operating systems' protection mechanism. Two important concepts in DAC are:

- File and data ownership: Every object in a system must have an owner. The access policy is determined by the owner of the resource. Theoretically, an object without an owner is left unprotected. Normally, the owner of a resource is the person who created the resource.
- Access rights and permissions: These are the controls that an owner can assign to individual users or groups for specific resources.

The main control permissions are control and control with passing ability. In order for a user who already has access permission to an object, to be able to assign or modify the object access permission, four basic models for DAC control exist: hierarchical, concept of ownership, laissez-faire, and centralized. A wide range of access modes is available in various DAC mechanisms such as read, write-append, write-update, write-change, write etc. Current operating systems have attempted to represent the aforementioned information using five basic mechanisms Capabilities, Profiles, Access control lists, Protection Bits and Passwords.

Mandatory Access Control

Mandatory Access Control (MAC) [8, 9] security labels or classifications are assigned to system resources that allow access only to entities (people, processes, devices) with distinct levels of authorization or clearance. These controls are enforced by the operating system or security kernel. The administrator manages access controls. The administrator defines a policy, which users cannot modify. This policy indicates which subject has access to which object. This access control model can increase the level of security, because it is based on a policy that does not allow any operation not explicitly authorized by an administrator. Two methods are commonly used for applying mandatory access control:

- Rule-based access controls: This type of control further defines specific conditions for access to a requested object. All MAC-based systems implement a simple form of rule-based access control to determine whether access should be granted or denied by matching:
 - An object's sensitivity label

- An subject's sensitivity label
- Lattice-based access controls: These can be used for complex access control decisions involving multiple objects or subjects. A lattice model is a mathematical structure that defines greatest lower-bound and least upper-bound values for a pair of elements, such as a subject and an object.

Role-Based Access Control

Role-Based Access Control (RBAC) [10, 11, 12] emerged in the 90s as an alternative technology to traditional discretionary and mandatory access controls for managing and enforcing security in large-scale enterprise wide systems. Its basic notion is that permissions are associated with roles and users are assigned to appropriate roles. It ensures that only authorized users are given access to certain data or resources. It also supports three well-known security principles: information hiding, least-privilege, and separation of duties. A role is a semantic construct forming the basis of access control policy. It is essentially a collection of permissions, and all users receive permissions only through the roles to which they are assigned.

Role hierarchy in RBAC is a natural way of organizing roles to reflect the organization's lines of authority and responsibility. Inheritance is reflexive because a role inherits its own permissions, transitive because of a natural requirement in this context, and anti-symmetry rules out roles that inherit from one another, and would therefore be redundant. Another useful extension is to allow constraints to be associated with role activation and resource access. Constraints may involve time, database queries and, for role activation, may require a principal to already be active in some set of prerequisite roles. RBAC models support constraints thus have a dynamic notion of role activation compared to the static principal-role assignment functions in simpler RBAC models.

RBAC standard [13] is organized into two main parts:

- the RBAC Reference Model defines a common vocabulary of terms for use in consistently specifying requirements and to set the scope of the RBAC features included in the standard
- the RBAC Functional Specification defines requirements over administrative operations for the creation and maintenance of RBAC element sets and relations; administrative review functions for performing administrative queries; system functions for creating and managing RBAC attributes on user sessions and making access control decisions.

10.1.1.2 Privacy policy languages

Platform for Privacy Preferences Project

The Platform for Privacy Preferences Project (P3P) is the first and most important example of a labelling protocol, a mechanism through which users can be informed of data requests and their consequences [3]. P3P, developed by the World Wide Web Consortium, is emerging as an industry standard providing a simple, automated way for users to gain more control over the use of personal information on Web sites they visit [14]. At its most basic level, P3P is a standardized set of multiple-choice questions, covering all the major aspects of a Web site's privacy policies. Their purpose is to present a clear snapshot of how a site handles personal information about users. P3P-enabled Web sites make this information available in a standard, machine-readable format. These P3P specifications can be checked by a Web browser or user agent, against user-specified preferences, to determine whether the organization follows user-acceptable privacy practices. This process is sometimes automated through software in a question-answer format. P3P enhances user control by putting privacy policies where users can find them, in a form users can understand, and, most importantly, enables users to act on what they see.

Technically, P3P consists of an XML vocabulary, a strongly defined set of base data types, and a rule-based language that acts on a set of rules used to express a user's preferences. Web sites express their privacy practices by means of a policy. Such policies consist of a static document, containing the identity of the organization responsible for the site, and a machine-readable text-based description of their privacy practices. When a site sends its P3P policy, the user-agent will verify that policy against the user's expressed preferences. On that basis the policy may be accepted or the user prompted to reject it.

P3P also provides a mechanism for specifying cookie-related privacy practices via compact policies. These compact policies are optional and simply offer a possibility for optimization by allowing user agents to check the privacy policies regarding a page's cookies without loading a separate policy document. Compact Policies are included in the HTTP response headers for a given webpage, and they allow a user agent to quickly assess the cookie-related policies of the page being loaded and respond accordingly. If an agent cannot glean the necessary information from the compact policy, then it may refer to the full P3P.

An additional element of the P3P work is A P3P Preference Exchange Language (APPEL) [14]. APPEL is a language for describing collections of preferences regarding P3P policies between P3P agents, although it is not needed to support negotiation of policies. Using this language, a user can express preferences in a set of preference-rules (called a ruleset), which can then be used by the user agent to make automated or semi-automated decisions regarding the acceptability of machine-readable privacy policies from P3P enabled Web sites.

Enterprise Privacy Authorization Language

EXTensible Access Control Markup Language (XACML) is a declarative, XML-based, access control policy language designed to express security policies and access rights to information for web services, digital rights management and enterprise security applications, that has been standardized in OASIS (Organization for the Advancement of Structured Information Standards) [16]. XACML describes both an access control policy language and a request/response language. The policy language is used to express access control policies. The request/response language expresses queries about whether a particular access should be allowed and describes answers to those queries. XACML as well as EPAL follow an abstract model for policy enforcement defined by IETF and ISO [17, 18, 19]. All requests for access to a protected resource go through an abstract component called a Policy Enforcement Point, or "PEP" (Access Enforcement Function in the ISO standard). The PEP formulates a request for an authorization decision that includes a description of the request in terms of identity, resource, purpose and operation. This authorization decision request is sent to another abstract component called a Policy Decision Point, or "PDP" (Access Decision Function in the ISO standard). The PDP retrieves the policies applicable to the request, along with any additional information required to evaluate those policies, evaluates them and the information available, and returns an authorization decision the PEP.

In XACML the authorization decision can be one of Permit, Deny, Indeterminate or NotApplicable. NotApplicable means the PDP is unable to provide an authorization decision because it has no policies that apply to the information provided in the request. If the PDP is unable to evaluate the policies or to retrieve required information for some reason, it will return an error instead of an authorization decision. Based on the authorization decision, the PEP either grants the requested access to the resource or denies access.

10.1.1.3 Identity management

The implementation of an Identity Management System (IDM) could be another solution in the protection of privacy at middleware level. An identity management system provides the tools for managing partial identities in the digital world. Partial identities are subsets of attributes representing a user, depending on the situation and the context. Each person may want to decide which partial identity to use in his relationship to the communication partner. Sometimes different names, either nicknames or pseudonyms are bound to the chosen partial identity. In general there are differentiated choices between the states of anonymity and identifiability depending on the user's wishes and the communication partner's prerequisite that must be supported by an identity management system. Identity management systems must support and integrate both techniques for anonymity and authenticity in order to control pseudonymity and liability of users. The first one refers to the lack of correlation between pseudonyms and their holders, meaning that the linkage of a pseudonym and its holder is not publicly known. This particular characteristic depends on their use in different contexts. If the same pseudonym is used in many cases, the corresponding data about the holder, which is disclosed through each use, can be linked. On the other hand the liability of a user must be controlled. A pseudonym can be authenticated in a secure way and based on this it

can be used to authorize the use of specific services. When necessary the holder of the pseudonym can be revealed and is liable for actions performed under this pseudonym.

In accordance to [20], identity management systems can be distinguished into three categories:

- identity management systems for account management, implementing an AAA-infrastructure (authentication, authorization and accounting),
- identity management systems for profiling of user data by an organization e.g. data warehouses which support personalized services or the analysis of customer behaviour,
- identity management systems for user-controlled context-dependent role and pseudonym management.

The user-side identity management systems can be classified based on the identity model in use [21]:

- Isolated user identity model: In this model the service providers are used for the distribution of both user identifiers and user credentials. It is the most common identity model in use; each service provider has its domain name and in most cases the users are identified by a user name and a password.
- Federated user identity model: In this model a user can be identified by all the service providers placed into the federation domain. This is based on agreements between the service providers, resulting to a user identified into one service provider domain to become identifiable, based on specified policies, by the other service providers within the federation domain. So, a mapping between the identifiers across the various service provider domains is required.
- Centralized user identity model: In this model the identity management is provided by a centralized entity. It can be further separated into the following categories:
 - Common user identity model: A separate entity is used for the provision of identifiers and credentials to the service providers.
 - Meta user identity model: The service providers' specific identifiers are mapped to a common meta-identifier.
 - Single Sign-On (SSO) identity model: Using this identity model, the user needs to sign-on (i.e. authenticate himself) once to get access to provided services. A separate single entity is used for identifiers allocation, credentials issuing and performing the actual authentication.
- User-centric user identity model: The users store identifier and credentials into a hardware tamper resistance device, a smart card or any other portable personal device, called Personal Authentication Device (PAD). This user-centric identity model can be combined with all the previously mentioned identity management models. The user only needs to authenticate himself to the personal device and then the device take the role of user into the authentication process with the service providers.

10.1.2 Solutions for middleware/application native privacy threats

10.1.2.1 Mobile agents security

Generally, Mobile Agents systems rely on a common set of baseline assumptions concerning security [22]:

- The Mobile Agent trusts the home platform where it is instantiated and begins its execution.
- The home platform and other equally trusted platforms are implemented securely.
- Public key cryptography, primarily in the form of digital signature, is utilized through certificates and revocation list managed through a Public Key Infrastructure.

Protecting the platform

The problem of protecting Mobile Agent platforms from Mobile Agents' malicious behaviours requires the performance of security checks both when a Mobile Agent arrives at the platform and during the execution phase. Before as well as during the execution of a Mobile Agent, the hosting platform should guard against a potential set of malicious instructions that may violate the platform's security policy and harm the platform's resources or operations.

A fundamental measure of defence against Mobile Agents with malicious intentions is access control. An access policy defines the rules for the authentication and the authorization of entities that request to access a system's resources. Authentication mechanisms are required for associating Mobile Agents with responsible owning entities, while access privileges are assigned to Mobile Agents by authorization means.

Access control is usually deployed with the use of cryptographic means. The authenticity, integrity and origin of a Mobile Agent may be ensured by a digital signature [23]. The author or the owner of the Mobile Agent digitally signs its code as an indication of the authority under which the Mobile Agent operates. The hosting platform enforces its security policy, deciding whether a Mobile Agent carrying a specific digital signature should be executed and moreover the execution privileges that should be assigned to the Mobile Agent. Several Mobile Agent systems use digital signatures for the confirmation of authenticity and integrity of Mobile Agents. Digital signatures benefit greatly from the availability of a Public Key Infrastructure.

There are several approaches to protecting a platform from malicious Mobile Agents, for example: RBAC based approaches [24, 25, 26], so-called Sandboxing [27], State Appraisal [28], Proof-Carrying Code [29], Path Histories [30], phRBAC [31] and many others.

Protecting the mobile agent

The security mechanisms for the protection of Mobile Agents against hosting platforms with malicious intentions can be classified according to whether they aim at the detection or the prevention of illicit manipulation. The former enables a Mobile Agent's owning entity to identify the occurrence of an attack; this is very useful for the validation of computational results of the Mobile Agent execution, the prevention of misuse of intercepted data and the identification of malicious hosts. Prevention security mechanisms strive to make it infeasible or useless for a hosting platform to attack a Mobile Agent during its execution. However, the full protection of a Mobile Agent during its execution is a very difficult and elusive problem.

There are several approaches to protecting a the Mobile Agents from hosting malicious platforms, for example: Partial Result Authentication Codes [32], Message Authentication Code [33], Execution Tracing [34], Oblivious Hashing [35], Environmental Key Generation [36], Code Obfuscation [37], Computing with Encrypted Functions [38], and others.

10.1.2.2 Web services security

WS-Security

WS-Security specification from OASIS [39] describes extensions to the SOAP messaging framework in order to accommodate security features such as message integrity, message encryption and message-based authentication. WS-Security builds on the extension mechanism of SOAP and adds abstract constructs that can introduce concrete security protocols on every SOAP message (Kerberos credentials or X.509 certificates are examples of known security tokens that can work with WS-Security). This standard utilizes two other well known security standards, XML-encrypt [40], a W3C Recommendation from W3C that specifies a process for encrypting data and representing the result in XML, and XML-Signature [41], a W3C Recommendation that specifies XML digital signature processing rules and syntax.

Web Services Trust Language

An extension to WS-Security is Web Services Trust Language [42] from IBM, BEA, Microsoft and others. WS-Trust is a language that provides additional mechanisms and extensions for security token exchange to enable the issuance and dissemination of credentials within different trust domains. WS-Trust complements the WS-Security specification in the sense that the WS-Security specification defines the way for two parties to exchange and assert security credentials while the

WS-Trust specification defines the mechanisms so that each party can determine if they can trust the asserted credentials.

Web Services Policy Language

A related specification in the area of web service security is the Web Service Policy Language (WSPL) specification from the OASIS group. WSPL is a subset of the eXtensible Access Control Markup Language (XACML), an XML syntax to express access control and role based policies. WSPL is defined as the "web service profile" of XACML and it provides the associated semantics for finding the intersection between any two WSPL policies (e.g. to find a mutually compatible policy between a client and a service). [43]. WSPL is the effort to introduce to the world of web services a standardized syntax to express policy rules related to security, access control and privacy.

Web Services Policy

The Web Service Policy (WS-Policy) specification provides an XML syntax to describe and communicate the policies of a web service. WS-Policy provides the means to find intersection between expressed assertions, so that the negotiations result in acceptance or deny verdict from the side of a client. WS-Policy is in fact a set of three standards, WS-Policy which defines the framework and an abstract policy model, WS-PolicyAttachment which defines two general-purpose mechanisms for associating policies with the subjects to which they apply, and WS-PolicyAssertions which defines a set of common message policy assertions that can be defined in a policy. The three specifications are available in [44, 45, 46].

10.1.2.3 CORBA security

The OMG Group (<http://www.omg.org/>) provided the CORBA Security Service Specification [47] in order to provide security features in the ORB environments. Within this specification a set of objects and their relationships are defined with the goal to provide identification and authentication, access control, secure communication between objects, non-repudiation, audit and security management services complying thus with the fundamental requirements of secure distributed systems which are confidentiality, integrity and accountability.

The structural model of CORBA Security has four major levels used during object invocation:

- Application-level components, which may or may not be aware of security;
- Components implementing the Security services, independently of any specific underlying security technology. These components are:
 - The ORB core and the ORB services it uses.
 - Security services.
 - Policy objects used by these to enforce the Security Policy.
- Components implementing specific security technology.
- Basic protection and communication generally provided by a combination of hardware and operating system mechanisms.

There are several implementations of the CORBA security standard: MICOSec (<http://www.micossec.org>), ORBAsec SL3 (<http://www.adiron.com/ORBAsec3.html>), etc.

10.1.2.4 RMI security

Many solutions to the security issues of the RMI architecture have been proposed in the literature, mainly focusing on authentication and authorization, for example:

- In [48] an extension to JMI security, based on Kerberos authentication system, Generic Security Service and Java Authentication and Authorization Service, is proposed and the performance of the proposed solution against the provided security level is studied. The proposed solution provides three security services: Discovery Service, Authentication Service and Dispatcher.

- In [49] a RMI security architecture that achieves mutual authentication between the client and the service, service access control and protected communication between the proxy and the service is proposed. The service authentication is split into two steps:
 - Firstly, a signed authentication proxy is introduced and is contacted by service requests, in order the clients to authenticate the proxy.
 - After the client authentication, a new client-specific session is created on the service. This service session captures the client authentication data and a proxy of the service is returned to the client. The authentication data can be reused by future remote method invocations in order to eliminate the need to repeat the authentication procedure.
- Concerning the special needs of some applications (e.g. generic logging, client side caching, security support) changes should be made to the default RMI behaviour at layers deeper than the application layer. A proposed approach is the use of smart proxies and interceptors with RMI [50]. The proposed framework utilises the Dynamic Proxy API while minimizes the changed needed to be made in the code. More specifically, no changes are needed on client side and only minimal changes are required on service side. The stub is replaced by a dynamic proxy; when a client tries to bind with a Remote object, instead of the stub it receives a dynamic proxy, together with an invocation handler and other required objects, like interceptors.

10.1.2.5 Publish-subscribe systems security

Specific methods addressing separately each one of the security and privacy issues (user anonymity, authentication, access control and mutual trust) have been proposed in the literature, for example:

- In [51], the security risks in content-based publish/subscribe systems are examined. Where it is possible to overcome these risks with current technology, the solutions are briefly described. In case of need for innovation the goal and the requirements of possible solutions are described. End-to-end authentication can be provided outside the publish/subscribe system, using commonly-used authentication technologies. Point-to-point authentication can replace the end-to-end authentication in the case that the publish/subscribe infrastructure is trusted. In this case, standard technologies can be used.
- For user anonymity provision, various techniques are proposed to be used in distributed systems. One example is onion routing anonymous connections. They are bidirectional and near real-time, and can be used instead of standard socket connections in proxy-enabled applications without the need for application modifications. The publish/subscribe routing and forwarding mechanism can also be used as a lightweight anonymity tool. For example, in the SIENA system [52], the used routing and forwarding algorithm requires in every hop only the knowledge of predecessor and successor hops and so any path that includes two or more intermediate hops is anonymous.
- In [53], in the context of the IST DELIS project, a solution for anonymizing users as well as hiding information on system dynamics and delivery of messages through anonymous paths has been proposed. For user anonymity provision, the publish/subscribe infrastructure has been modified by the introduction of an anonymity layer. The proposed solution is based on universal re-encryption, anonymous communication based on URE-Onions and signatures for cipher texts that can be universally re-encrypted.
- In [54], a method to specify access control policy rules using expressions similar to subscription expressions is described. These policies define access rules for publish and subscribe functions and screening rules for notifications. With the assumption that subscribers and publishers trust the local infrastructure, only positive access rights are granted. A user without the appropriate access rights cannot subscribe for or publish any event. Subscription and advertisement filters are applied as well as covering relations to define the access rights.

- In [55], a secure publish/subscribe architecture is proposed. This architecture addresses initial authentication and maintenance of identity, scalable topic security, and message-level security that protects messages over multiple hops with varying underlying transport security.
- In [56], an integration of the OASIS role-based access control (RBAC) [57] into the HERMES publish/subscribe system [58] is studied. The system supports many advanced features, such as the ability to work within a network where nodes are attributed different levels of trust. In [59], groups of trust are devised to model and implement security constraints both on the application and the system level. The concept of scopes (implemented in the REBECA publish/subscribe system [60]) helps to localize and implement security policies as an aspect of structured publish/subscribe systems.

10.1.2.6 Peer-to-peer system security

The main research in P2P systems, related to security and privacy issues, has been focused in user anonymity, for many reasons, such as distrust to governments. Nowadays, there are many anonymous P2P networks that differ in the kind of anonymity they provide, for example: ANts P2P network [61], Freenet [62], Entropy [63], GUNet [64], I2P [65], MUTE [66], Nodezilla [67], Rodi [68], Share [69], Sumi [70], TOR [71], DirectConnect [72], WASTE RSA-secured system [73], etc.

Trust management in P2P systems can be classified into three categories:

- In credential and policy-based trust management systems, peers use credential verification to establish a trust relationship with other peers. Since the primary goal of such systems is to enable access control, their concept of trust management is limited to verifying credentials and restricting access to resource according to application-defined policies. PolicyMaker [74] is a characteristic example; each peer can specify its policies locally and may grant another peer access to its service if the providing peer can determine that the requesting peer's credentials satisfy the policies.
- In reputation-based trust management systems a peer requesting a resource may evaluate the trust in the reliability of the resource and the peer providing the resource. Peers in such systems establish trust relationships with other peers and assign trust values to these relationships. Trust value assigned to a trust relationship is a function of the combination of the peer's global reputation and the evaluating peer's perception of that peer. Reputation-based P2P systems include SPORAS [75], HISTOS [75], XRep [76], NICE [77], DCRC/CORC [78], P2Prep [79] and EigenRep [80].
- In social network-based trust management systems the social relationships between peers are utilized when computing trust and reputation values. In particular, these systems form conclusions about peers through analyzing a social network that represents the relationships within a community. Regret [81] and NodeRanking [82] are the most characteristic social network-based trust management systems.

In [83], the integration of policy-based and reputation-based trust management approaches into a versatile trust management language, called PROTUNE, capable of addressing both the structured organizational environments as well as unstructured user communities is proposed.

10.2 Privacy in the network

10.2.1 Anonymous networking

10.2.1.1 Chaum's MIX

The solutions currently being developed worldwide to counter traffic analysis attacks are based on David L. Chaum's mix node idea [84]. A mix node is a computer which sends messages generated by users and that processes each message before delivering it. The mix node's purpose is to hide the correspondence between the mail received by the mix node and the mail that it retransmits to

the final destination. When a user decides to hide the destination of his mail, he encrypts the message, including the mail header, with the RSA public key of the mix and then wraps it in a new packet with the mix address as destination. Each mix node decrypts the messages using its private key, waits for the arrival of a fixed number of messages, and then rearranges all the messages in lexicographical order before transmitting them.

We can state that the techniques used nowadays to counter traffic analysis attacks are basically as follows:

- Using Public Key Cryptography to recursively wrap messages.
- Employment of elements which in turn decrypt, delay, and re-order messages before relaying them onward.

10.2.1.2 Traffic analysis countermeasure

All traffic analysis countermeasures mainly deal with three parameters:

- packet format
- packet routing algorithm
- packet forward methods

While the format of the packets is almost the same for all anonymity systems, the choices deployed for packet routing and forwarding techniques strongly depend on the typology of traffic that the system is planned to support. Packet length countermeasures consist of padding data to obtain fixed length packets. In this way an attacker is unable to gain information from the packet size.

The basic idea as far as packet routing is concerned is that, once padded, the packet is not sent directly to its destination, but routed through a MIX network in order to hide its path and destination. The packet is thus wrapped by adding the identity of all the mix nodes it has to cross to its header and recursively ciphering it with the public key of these mix nodes. In this way, each mix node is able to unwrap the message using its own private key and to send it to the next hop along the path. Currently, the decision about the path followed by the packet can be made in advance by the user or can be chosen hop by hop, in a random way. Finally, the aim of different packet forwarding techniques is to hide traffic behaviour from the external observer. There are many approaches which can be used to forward messages:

- Message Threshold: the mix nodes wait until they receive a certain number of messages before forwarding all of them at the same time, thus modifying the order of arrival.
- Message Pool: this is the hashing algorithm for mixmaster [85]. It has two parameters: the pool size and the probability of sending. The nodes wait until they have enough messages in their pool, and then they forward each message in the queue with probability "p".

The anonymity systems developed until now can be classified in two different group:

- The Large Latency System provides a high privacy level but is unable to support real time traffic. The most widely used system in this category, Mixminion, will be analyzed in the following section.
- Low Latency system which are thought to support real time traffic. Tor and Tarzan represent the most used systems belonging to this group.

10.2.1.3 TOR

Tor (The second generation Onion Routing) [86] is a circuit-based low-latency anonymous communication service. This system is based on the first generation Onion Routing [87][88], a distributed overlay network designed to anonymize TCP-based applications such as web browsing, secure shell, and instant messaging. In the Onion Routing system, clients choose a path through the network, building a circuit in which each node (or "onion router" or "OR") in the path knows its predecessor and successor, but none of the other nodes in the circuit. Traffic flows down the circuit

in fixed-size cells which are unwrapped by a symmetric key at each node, like the layers of an onion, and relayed downstream.

The Tor network is an overlay network in which each onion router (OR) runs as a normal user-level process without any special privileges. Each onion router maintains a TLS [89] connection to every other onion router. Each user runs local software called an onion proxy (OP) to fetch directories, to establish circuits across the network, and to handle connections from user applications. These onion proxies accept TCP streams and multiplex them across the circuits. The onion router on the other side of the circuit connects to the requested destinations and relays the data. Each onion router maintains a long-term identity key which it uses to sign TLS certificates and router descriptors (a summary of its keys, address, bandwidth, exit policy, and so on), and a short-term onion key. This second key is used to decrypt requests from users to set up a circuit and negotiate ephemeral keys.

10.2.1.4 MixMinion

MixMinion [90] is a message-based anonymous remailer protocol with a different path for reply messages. Mix nodes cannot distinguish MixMinion forward messages from reply messages, so forward and reply messages share the same anonymity set. MixMinion works in a real-world Internet environment, requires little synchronization or coordination between nodes, and protects against known anonymity-breaking attacks. MixMinion is considered a Type III Anonymous remailer protocol. This system, in fact, is based on previous remailer systems. The first implementation of mix based remailers was produced by contributors to the Cypherpunks mailing list [91]. These Type I anonymous remailers were inspired by the problems that surrounded the anon.penet.fi anonymous forwarding service [92], and by theoretical work on mixes. Two of the group's founders of Cypherpunks, Eric Hughes and Hal Finney, built a secure anonymous mail system for the Internet that used Phil Zimmermann's PGP to encrypt and decrypt remailed messages sent through the mix chain. Later on, Cottrell implemented the Mixmaster system [85, 93], known also as Type II remailer, which added message padding, message pools, and other mix features which were lacking in the Cypherpunk remailers.

MixMinion has been developed to provide anonymity against a well-funded adversary who can observe all traffic on the network, generate, modify, delete or delay traffic on the network, operate mixes of its own and compromise some fraction of the mixes on the network.

10.2.1.5 Tarzan

Tarzan [94] is a peer-to-peer anonymous IP network overlay. Working at the IP layer, Tarzan is general-purpose and transparent to applications. Tarzan achieves its anonymity with layered encryption and multi-hop routing, in a similar way to a Chaumian mix. A message initiator pseudo-randomly chooses a path of peers through a restricted topology. Cover traffic (dummy packets) prevents a global observer from using traffic analysis to identify an initiator. Tarzan uses a Network Address Translator (NAT) to bridge between Tarzan hosts and Tarzan unaware Internet hosts. Tarzan aims to prevent attacks from a global eavesdropper. An adversary observing the entire network should be unable to determine which Tarzan relay initiates a particular message. All Tarzan nodes run software that discovers other participating nodes, intercepts packets generated by local applications that should be anonymized, manages tunnels through chains of other nodes to anonymize these packets, forwards packets to implement other nodes' tunnels, and operates a NAT (network address translator) to forward other participants' packets onto the ordinary Internet.

10.2.2 Wireless networks enhancement

10.2.2.1 Approaches to protect privacy in wireless network

There are two broad approaches to protect information privacy: access control and anonymisation. Briefly, access control can be used to protect information privacy by building an architecture which allows one actor to restrict the ability of other actors to retrieve information. Several techniques have been developed, access control matrix, role based access control, multi-subject multi-target

policies, encryption and digital certificates. Anonymity, defined as “the state of being anonymous” or “having no name”, can be used to protect privacy by ensuring that any information released to an untrusted party cannot be associated with a real-world identity.

While access control has been the dominant mode of enabling location privacy so far, there are a combination of factors which make anonymity-based techniques more attractive than traditional access control methodologies, for example user may better accept anonymized location service while may be reluctant to accept service which require disclosure of real-world identity. Configuration of access rights is a difficult task. If an application works well with anonymous location data, it needs not be trusted.

10.2.2.2 Decrease wireless privacy threats

There are several methods to reduce low layers privacy threats. They are based on anonymity in such a way eavesdropper and LBS provider cannot determine the originator of the message, for example:

- As mobile device use by default a fixed MAC address, it is quite easy tracking device location over time and so user's movements. In [95] a solution is proposed, the MAC address of device is changed on every association with an access point. An algorithm generates disposable MAC address and it is foreseen a method to detect duplicate address. The goals of proposed solution are to offer unlinkable identifier, minimal network disruption and applicability.
- High precision tracking systems combined with new correlation attacks, which utilize spatial and temporal correlation between old and new address, can defeat privacy level reached by solutions as previous. In [96] the concept of silent period is proposed to combat such correlation attacks. The silent period is defined as a transition period between using new and old pseudonyms in which a station is not allowed to disclose either the old or the new pseudonym. As a result, silent period disrupts the temporal and spatial correlation between two separately received pseudonyms and obscures the time and the place where a pseudonym changed, this makes more difficult to associate two different pseudonyms with the same mobile node.
- Solving the linkability problem is the objective of the work in [97] where a framework is proposed based on the concept of transaction. Traffic exchange occurring within the same transaction is linkable, whereas different transactions are unlinkable with each others. This is achieved through appropriate registration procedures and the introduction of random delay between different transactions.

There are several methods to reduce low layers privacy threats, for example:

- To reduce threat from high layers, [98] proposed an approach in which location-based services collect and use only de-personalized data, that is, practically anonymous [99]. This approach is beneficent for both parties. For service provider anonymous data cause less overhead, in fact data can be even distributed to third parties without user consent. For data subjects it removes the need to evaluate complex service provider privacy policy. The algorithm uses a centralized broker service that receives messages from mobile nodes, de-personalizes them, applies an algorithm to reduce spatial or temporal resolution, and then forward them to external service provider.
- A disclosure-control algorithm is presented in [100], in which areas are classified as either sensitive or insensitive for the user point of view. The algorithm works minimizing position inaccuracy, for third party applications, when the user is located in an insensitive area, and maximizing position inaccuracy when the user is located in a sensitive area.
- A privacy awareness system (PawS) is proposed in [100] for ubiquitous computing environments. PawS aims at providing users with tools that let them protect their personal privacy and help others respect that privacy. It is based on respect and social and legal norms rather than rigorous technical protection of private information. In pawS, when a user enters an environment in which services are collecting data, a privacy beacon announces the

privacy policies of each service in the environment. A user's privacy proxy checks these policies, against the user's predefined privacy preferences.

- An integrated approach is proposed by [101]. Their scheme, called Caravan, is applied to a VANET (Vehicular Ad hoc NETWORK), but can be easily considered even in other types of wireless networks. In order to provide unlinkability to mobile nodes, Caravan makes use of silent period; moreover, it introduces the "group" concept. This avoids overhearing pseudonyms. Within a group there is a group leader that communicates on behalf of the group, so the remaining mobile nodes can remain in silence for an extended period of time, bounded by the time they remain in the group. Employing the group concept, since not all mobile nodes must broadcast messages, the level of anonymity can increase.

10.2.2.3 Sensor networks enhancement

Sensor networks raise significant security and privacy problems. To solve them, solutions have been introduced in the recent past. For what concern security issues, there are several solutions, in different areas, as follows:

- Key establishment and trust setup. When setting up a sensor network, one of the first requirements is to establish cryptographic keys for later use. There are three types of general key agreement schemes:
 - trusted-server scheme depends on a trusted server for key agreement between nodes and therefore is not suitable for sensor networks. In fact, there is usually no trusted infrastructure in sensor networks.
 - self-enforcing scheme depends on asymmetric cryptography, such as key agreement using public key certificates. However, limited computation and energy resources characterizing sensor nodes often make it undesirable to use public key algorithms.
 - key pre-distribution scheme, where key information is distributed among all sensor nodes prior to deployment. Following that approach, [102] proposes a novel random key pre-distribution scheme that exploits deployment knowledge and avoids unnecessary key assignments.
- Secrecy and authentication. Like traditional networks, most sensor network applications require protection against eavesdropping, injection, and modification of packets. Cryptography is the standard defence. The large number of communicating nodes makes end-to-end encryption usually impractical since sensor node hardware can rarely store a large number of unique encryption keys. Alternatively one may opt for hop-by-hop encryption, in which each sensor node stores only encryption keys shared with its immediate neighbours.
- Secure routing and robustness to communication denial of service. Routing and data forwarding is an essential service for enabling communication in sensor networks, unfortunately, current routing protocols suffer from many security vulnerabilities [103]. Moreover adversaries can severely limit the value of a wireless sensor network through denial-of-service attacks.

10.2.3 RFID systems

10.2.3.1 The regulation approach

This section does not offer an exhaustive discussion of the approach since it is addressed in another chapter concerning the legal aspects of privacy. In brief, this solution is able to impose a restricting framework for manufacturers using RFID tags. Consumer associations have already voiced their concerns to the relevant authorities. Certain solutions, such as a label indicating that the item contains a RFID or the killing of each tag at the exit of the store, appear to be necessary.

10.2.3.2 The “kill” tag approach

This solution consists of creating a new command, the “kill” command that enables the destruction of the tag which, subsequently, cannot be re-activated [104, 105]. It is possible to deactivate the tag via a fuse linked to the chip's power supply or antenna, or even by erasing its memory. This command should be performed at the checkout of the store. This approach, defended by consumer associations, solves all the problems of privacy. However, outside the supply chain, all the advantages of RFID disappear. Thus, the future role of Ambient Intelligence is scaled down: there is an end to the possibility of designing fridges which talk to food, or washing machines which communicate with clothes. Even the control of part of the supply chain is removed. The tracking of items for recycling or for after-sale services is no longer feasible. With such an approach, RFID remains merely a tool to improve the logistics of manufacturers and distributors.

10.2.3.3 The Faraday Cage approach

Another basic solution is to enclose the RFID tags in a wallet made of a metallic sheet or mesh [106]. This wallet plays the part of a Faraday cage, blocking the HF and UHF radio signals of readers. The solution surely works but this approach is impractical for large tagged objects that cannot be placed in containers easily. A further problem is raised by the fact that the RFID tags will be so small in the near future that we will not know where they are. As a consequence, it will be impossible to pack them.

10.2.3.4 The active jamming approach

It is possible to create a device that emits signals in the same bands as RFID readers, that is to say in 13.56 MHz and 2.4 GHz, in order to jam communication with RFID tags. This device would need to broadcast signals with a higher amplitude than the different standards permit and, consequently, would be illegal. Nevertheless, as shown in the next chapter, smart jamming is also possible.

10.2.3.5 Antenna energy analysis approach

This solution is based on the assertion that distance leads to distrust. The closer the reader is, the more subject it is to scrutiny by the wearers, owners or users of the tagged object. To enhance privacy, RFID tags might be able to employ the signal-to-noise ratio of the transmissions they receive from a reader to estimate the distance of that reader from the tag. Whether the tag responds or not to the reader requests depends on the distance [107]. This kind of approach is effective but destroys an important advantage of RFIDs, that of long distance reading.

10.2.3.6 The encryption approach

This approach uses cryptographic methods incorporated in the tag to protect privacy while providing the desired functionalities. This approach does not follow the trend of reducing the tag cost. Today, the target cost of a tag (antenna plus chip) is 5 cents. This does not seem feasible when considering the increasing number of transistors required for cryptographic algorithms. Three different techniques appear in the literature:

- The “Hash-Lock” approach: In this solution, a tag can be “locked” and therefore unable to answer its ID until it is unlocked [108]. The tag is locked by a code y , and it is only unlocked by the presentation of a key or PIN value x such that $y=h(x)$ for a standard one-way hash function h . The following example explains how it would work in a supermarket. A consumer provides a private code y (that of a loyalty card, for instance) for the tags at the checkout, and then transmits the unlocking PIN x with a specific reader with close coupling to unlock tags on returning home. This approach also requires a randomized hash function in order to protect items from tracking. The main drawback for the consumer is the waste of time unlocking his shopping basket.
- The re-encryption approach: This solution emerged when Juels and Pappu [109] tried to solve the privacy implications of tracking RFID tags on banknotes. Their banknotes are

encrypted with a law-enforcement public key. A periodic re-encryption of the cyphertexts is required to lower the risk of linkability of multiple appearances of the same tag. This encryption is provided by public privacy-enhancing readers in stores since each RFID tag has reduced computing resources. The cost for installing such a network of readers is the main problem to face.

- Silent tree-walking: This approach provides one answer to the risk of passive eavesdroppers hearing the signal broadcasted by the reader that may be picked up from a few hundreds of meters contrary to the signal transmitted by the tag that is really weaker. In deterministic singulation protocols (for example the tree-walking algorithm often used in the RFID standards to avoid different tags answering together), the reader calls tags using only a part of the UID and increases the number of bits until it only finds a single tag. Thus, by listening to the reader, we may learn a part of the UID of each tag. To prevent these attacks, the tags are able to generate their own random pseudo-IDs before singulation [108]. Nevertheless, the impact of the increase in the number of transistors has not been taken into account, above all in a scenario where manufacturers are making drastic efforts to reach a price of 5 cents for a tag.

10.2.3.7 The blocker tag approach

Jules et al [110] have developed a device which is capable of blocking a tree-walking singulation protocol, thereby preventing intrusive readers from knowing the tag's UIDs. To understand this interesting device, it is important to describe how tree-walking protocols work with the help of an example. If, for instance, the UIDs have only three bits, only eight tags are possible. Let us consider that tags `001`, `011` and `110` are present in the field. Firstly, the reader calls all the tags beginning with a `0`, two tags answer thereby creating a collision. To resolve it, the reader increases the number of bits and calls the tags beginning by `00` to separate the different tags. Only `001` answers: it is recognized by the reader. Secondly, the latter starts a recursion from the collision points trying other branches of the tree. That is to say, it calls `01`: `011` answers. Finally it calls `1` and `110` answers. The idea of the blocker tag, resembling a tag we can put in our pocket, is to emit both `0` and `1` thus creating a collision and forcing the reader to start its singulation algorithm. If the blocker tag emits `0` and `1` simultaneously (two antennas required) at each node of the tree, the reader may never complete its algorithm. As a consequence, the consumer is protected against unwanted scanning.

10.2.3.8 The Watchdog tag

Floerkemeier et al. [111] describe a device named "watchdog" tag that has the functionalities of a tag but may be incorporated in a mobile phone or a PDA. The "watchdog" tag is an audit system for RFID privacy that monitors the readers in the vicinity. Thus, if a reader scans your personal tags, the screen of the "watchdog" tag displays on the screen the intrusion attempt and the features of the reader. This kind of device can make the danger but it is ineffective of removing it.

10.2.3.9 Improvement of the standard

An improvement in privacy may simply be achieved by incorporating more information in the inventory request of the reader [111]. The data provided by the reader to the tags include the data collector ID, the policy ID and the reader ID. The data collector ID defines the purposes of the scan. The policy ID and reader ID enable acquisition of details regarding policy information for the network connected to the reader, its authentication and its approximate location. This approach is an easy way to solve the purpose specification (3), the use limitation (4), the openness (6) and the accountability (8) of the Fair Information Practice but imposing a new ISO standard requires a great deal of persuasion.

10.3 Keys management, authentication and accounting

10.3.1 Key management

The cornerstone of network security resides in the use of cryptographic keys. They are small pieces of information, used by encryption and decryption functions, which allow ensuring that no unauthorized entity can read and understand the original information. One of the major issues related to cryptographic keys is to secure their management in order to ensure their administration.

While keys management procedures, protocols and infrastructure can benefit from the stability of wired networks, their use and adaptation in the context of the increasing mobility and wireless technologies become in most cases inappropriate. In fact, mobile ad hoc and wireless sensors networks suffer from strong operational constraints (e.g. dynamic topologies, variable connectivity, computational or power limitations) which fundamentally impact previously existing approaches. This section introduces first the cryptography basis, terminology and definitions related to key management paradigm. Then, after briefly reminding the Internet Key Exchange (IKE) protocol, it surveys how keys are managed in mobile ad hoc and wireless sensors networks.

10.3.1.1 Cryptography terminology, definition and basis

Trusted third parties, Certificate authorities and Public key infrastructures

A Trusted Third Party (TTP) is an entity which enables interactions between two parties who both trust the third party; they use this trust to secure their own interactions. Examples of trusted third parties are Key Distribution Centres (KDC), Key Translation Centres (KTC) and Certificate Authorities (CAs). More specifically, a Certificate Authority is an entity which issues digital certificates for use by other parties.

CAs are commonly used in many Public Key Infrastructure (PKI) schemes. In fact, public key cryptography (see below) requires that the authenticity of the public keys can be established. A straightforward approach requires that any two users that wish to communicate must exchange their public keys in an authenticated manner. However, by having a trusted third party issue certificates to each of the users only the public key of the TTP needs to be distributed to each of the users. A Public Key Infrastructure enables management operations for such certificates.

Cryptographic algorithms and keys

Cryptographic algorithms can be classified in two major groups: symmetric and asymmetric key algorithms:

- Symmetric key algorithms use the same key for both encryption and decryption. They are usually faster to be executed, but require sharing the secret key between the sender and the receiver. Therefore, when communication has to be established for a group of nodes, the system usually faces scalability issues because each sender-receiver pair should share a key. Moreover, if a sub-set of the nodes group relies on a unique key, the global system becomes vulnerable if only one of those nodes is compromised.
- Asymmetric (or public) key algorithms rely on two different keys for encryption and decryption. They are based on some mathematical principles which make it infeasible or impossible to obtain one key from another. In this way, one of the keys can be made public while the other is kept secret (i.e. private). This is called public key cryptography.

Keys management approaches

The primary objective of key management systems is to share a key among a specified set of participants. As pointed by, [112] the main key management schemes are key pre-distribution, key arbitration, and key agreement:

- Key pre-distribution schemes aims at distributing keys to all the nodes before communication establishment. Such approach induces less communication and computation. However, all the nodes have to be known in advance. The usual major drawbacks of such schemes are that new members cannot be included in the group and that dynamic change of the key is not possible.

- In key arbitration schemes, a particular node, the arbitrator, creates and distributes keys among all participants. Such a scheme implies that the administrator node needs to be always reachable and to have sufficient computational and power resources. One possible variant of such scheme is to distribute the keying service. However, arbitrator replication procedures are expensive and introduce system vulnerabilities to attacks.
- Key agreement schemes are mainly based on asymmetric key algorithms. Such solutions aim at enabling the agreement on a secret key between two or more nodes. In group key agreement schemes, each participant contributes a part to the secret key. Usually such schemes rely on high computational complexity.

10.3.1.2 Key management in the Internet

In the Internet, key exchange is ensured by the Internet Key Exchange (IKE) protocol [113]. IKE is an Internet Protocol Security (IPSec) [114] standard protocol, which is able to ensure security for Virtual Private Network (VPN) negotiation and remote host or network access. It allows to automatically set up Security Associations (i.e. negotiation and authentication).

IKE is considered as an hybrid protocol due to the fact that it incorporates parts of the OAKLEY [115] and the SKEME [116] protocols within the Internet Security Association and Key Management Protocol (ISAKMP) [117] framework. Moreover, IKE is based on a Diffie-Hellman key exchange to establish a shared session secret, from which cryptographic keys are derived. Public key techniques or, alternatively, pre-shared secrets, are used to mutually authenticate the communicating parties.

Relying on ISAKMP framework, IKE operates according two phases:

- ISAKMP Phase 1: In this phase two ISAKMP peers negotiate a secure and authenticated communication channel. During this phase the two entities build an ISAKMP Security Association (SA).
- ISAKMP Phase 2: Once the ISAKMP SA established, SA are negotiated on behalf services such as IPSec.

While IKE is not required for IPSec configurations, it enables automatic negotiation and authentication, anti-replay services, support to CA and the ability to change encryption keys during an IPSec session. Note that IKEv2 expands upon IKEv1 and is defined in [118].

10.3.1.3 Key management in mobile ad hoc networks

As mentioned above, mobile ad hoc networks strongly impact usual security approaches. This is mainly due to the intrinsic nature of such communication system: host mobility, unreliable wireless media or lack of infrastructure. Moreover, the computational load and complexity for key management extremely suffer of available nodes resources and of the dynamic nature of network topology.

As summarized and expressed by [119], mobile ad hoc networks lack of three important kinds of infrastructure:

- Routing infrastructure, ensuring stable connectivity with all the nodes constituting the network;
- Services-related infrastructure, such as name resolution or TTPs;
- Administrative infrastructure, providing support to Cas.

In such a context, efficient keys management solutions are mandatory but become a real challenge. The following sections surveys recent approaches to solve keys management issues. These approaches can be classified in three categories: virtual certificate authorities, certificate chaining and composite approaches.

Virtual Certificate Authorities approaches

Virtual Certificate Authority (CA) is an approach which aims at securely distributing the CA functionality over multiple nodes. In this way, the virtual CA is compromised only if a certain fraction

of the key shares have been themselves compromised by an attacker. In such an approach, to get certification services, an end-user requires only access to a subset of the nodes involved in the virtual CA. This approach is usually based on threshold cryptography paradigm [120, 121].

Certificate chaining approaches

In order to mitigate the use of heavy and complex key management infrastructures the certificate chaining approach is used. In such approach, where every node plays the same role and has identical responsibilities, usually a set of digital certificates is required for the authentication service.

Composite key management

A more recent scheme combines the virtual CA and the certificates approaches as mentioned in [122, 123, 124] and in [125, 126, 127]. Such approach can be viewed as a trade-off between security and flexibility of the system by benefiting of the advantages of central and fully distributed trust models.

10.3.1.4 Key management in wireless sensors networks

In most cases, wireless sensor networks are composed by nodes characterized by computational capacity, memory space and power resource limitations. As for mobile ad hoc networks, wireless sensor networks are also vulnerable to various kinds of malicious attacks (e.g. eavesdropping or traffic-analysis). In such context, taken into consideration both the devices constraints and the complexity of existing key management systems, various solutions had been proposed. They can be classified in three categories: online server-based systems, public key-base key management and key pre-distribution solutions.

Online server-based systems

As mentioned previously, most of the key management solutions for mobile ad hoc networks are designed in order to decentralize the online key management. Even it would be also necessary for wireless sensors networks, as in mobile ad hoc networks, most of the above propositions rely on asymmetric cryptography. Due to sensors networks computational and power limitations, these kinds of system appear too expensive in the context of sensor network.

Nevertheless, symmetric key management is often complicated, even computational complexity or power consumption is negligible compared to public-key methods. Moreover, asymmetric key management can enable flexibility and scalability to the global sensor network. For these reasons, some approaches, in view to elaborate asymmetric key management systems, are focused on the way to offload certain computation to servers in order to reduce low-end devices operations.

Public key-based key management

As explained, public key cryptosystems had been first considered as too expensive and too heavy for being used in the context of wireless sensors networks. Nevertheless, later, several solutions revealed the potential possibility to decrease the computational costs related to public key management scheme. Most of these solutions rely on ECC-based key methods. Such protocols, such as [128, 129,130], appear well-suited for constrained mobile environments. It is mainly due to the property of small key sizes.

Key Pre-distribution solutions

Even public key cryptosystem solutions exist for wireless sensor networks, their full and concrete deployment face the limited computational and power capacities of sensor nodes. In this way, secret key pre-distribution can be considered as one of the practical approaches for establishing secure channels in such constrained networks.

10.3.2 Security services and AAA

Sometimes referred to as "triple-A" or just AAA, authentication, authorisation, and accounting is a framework providing support for Service Providers to manage and control users.

- Authentication provides a vehicle to identify a client that requires access to some system and logically precedes authorisation. The mechanism for authentication is typically

undertaken through the exchange of logical keys or certificates between the client and the server.

- Authorisation follows authentication and entails the process of determining whether the client is allowed to perform or request certain tasks or operations. Authorisation is therefore at the heart of policy administration.
- Accounting is the process of measuring resource consumption, allowing monitoring and reporting of events and usage for various purposes including billing, analysis, and ongoing policy management.

10.3.2.1 Radius

The Remote Access Dial In User Service (RADIUS) protocol was developed by Livingston Enterprises around 1989 and further improved by Merit University (Michigan). RADIUS is an Authentication Authorisation and Accounting (AAA) protocol based on client-server architecture. The client is a Network Access Server (NAS) which desires to authenticate and authorise its links. On the other hand, the server is an entity which has access to a database containing the ID of all the registered users together with authentication, authorisation and accounting information for each one of them. A RADIUS server can act as a proxy client to other RADIUS servers or even to other kinds of authentication servers.

10.3.2.2 TACACS+

Terminal Access Control Access System was the first protocol to allow a NAS to offload user authentication, authorisation and accounting to a central server. Both TACACS and its sequel XTACACS are missing many features that the latest version, called TACACS+, and RADIUS offer. Moreover, TACACS and XTACACAS have reached end-of-maintenance, and should no longer be deployed in new installations. Due to this, TACACS and XTACACS will not be discussed in this survey. TACACS+ mainly improves on TACACS and XTACACS by separating the functions of Authentication, Authorisation and Accounting and by encrypting all traffic between the NAS and the TACACS+ server. Moreover, TACACS+ allows any authentication scheme to be used, since it allows for arbitrary length and content exchanges. The protocol allows the NAS client to request very fine grained access control and allows the daemon to respond to each component of that request.

10.3.2.3 DIAMETER

The DIAMETER basic protocol is designed to provide a framework for services requiring AAA support, at the access technology level. The protocol is intended to be flexible enough to allow services to add building blocks (or extensions) to the base DIAMETER protocol to meet their requirements. Unlike other AAA protocols for access technologies - such as PPP dial-in, Mobile IP and others -, DIAMETER uses a peer to peer architecture rather than a more classic client/server scheme. DIAMETER is recognised as a peer to peer protocol since any node is free to initiate a request at any time. Messages initiated by a server towards a client are usually requests to abort a service to a specific user.

DIAMETER is also meant to operate both with local and with roaming situations. Since DIAMETER is not a complete protocol by itself, but it needs application-specific extensions from the technology, or architecture, used to access the network, it is not possible to describe or compare the protocol's details regarding security and other aspects. Thus, the following discussion will deal mainly with the elements that are provided by the basic common DIAMETER framework: message format, message transport, error reporting, accounting and security considerations. DIAMETER is still a draft from the Authentication Authorisation and Accounting IETF group.

10.4 References

- [1] E. D. Schoeman, "Philosophical Dimensions of Privacy: An Anthology," New York, NY, Cambridge Univ. Press, 1984. Informatics, vol. 49, no. 1, pp. 39-44, 1998.

-
- [2] H. Leino-Kilpi, M. Valimaki, T. Dassen, M. Gasull, C. Lemonidou, A. Scott and M. Arndt, "Privacy: A review of the literature," *International Journal of Nursing Studies*, vol. 38, pp. 663-671, 2001.
- [3] Mark S. Ackerman, "Privacy in pervasive environments: next generation labeling protocols." *Personal and Ubiquitous Computing*, 8(6): pp. 430-439, November 2004.
- [4] CSIS, "Security Glossary," Information Systems Security Organization, 2003. Online: http://ise.gmu.edu/~csis/glossary/merged_glossary.html
- [5] B. Lampson. "Protection", *Proc. of the 5th Symposium on Information Sciences and Systems*, Princeton, NJ, pp. 437-443, March 1974.
- [6] R. Sandhu, P. Samarati. *Access Control: Principles and Practice*, IEEE Communications Magazine, Volume 32 Issue 9, pp. 40-48, Sep. 1994.
- [7] Carole S. Jordan, National Computer Security Center "A guide to understanding discretionary access control in trusted systems", NCSC-TG-003-87, 1987.
- [8] Dorothy E. Denning. *A Lattice Model of Secure Information Flow*, *Communications of the ACM*, Volume 19 Issue 5, pp. 236-243, May 1976.
- [9] Ravi S. Sandhu. *Lattice-Based Access Control Models*, *IEEE Computer*, Volume 26 Issue 11, pp. 9-19, Nov. 1993.
- [10] David F. Ferraiolo and D. Richard Kuhn, "Role-Based Access Controls", *Proceedings of the 15th NIST-NSA National Computer Security Conference*, Baltimore, Maryland, October 13-16, 1992.
- [11] David F. Ferraiolo, Janet A. Cugini, and D. Richard Kuhn, "Role-Based Access Control (RBAC): Features and Motivations", *11th Annual Computer Security Applications Proceedings*, 1995.
- [12] Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E. "Role Based Access Control Models". *IEEE Computer* 29, 2 (Feb 1996), pp. 38-47.
- [13] ANSI/INCITS 359-2004 *Information Technology - Role Based Access Control International Committee for Information Technology Standards*.
- [14] Privacy and P3P/ APPEL. Online: <http://www.w3.org>
- [15] Enterprise Privacy Authorization Language Online: <http://www.zurich.ibm.com>
- [16] Organization for the Advancement of Structured Information Standards Online: www.oasis-open.org
- [17] A. Westerinen, et al., "Terminology for Policy-Based Management", IETF RFC 3198, November 2001. Online: <http://www.ietf.org/rfc/rfc3198.txt>
- [18] *A Framework for Policy-based Admission Control*, by R. Yavatkar, et al., IETF RFC 2753, January 2000. Online: www.ietf.org/rfc/rfc2753.txt
- [19] ISO/IEC 10181-3:1966 *Information technology, Open Systems Interconnection, Security frameworks for open systems: Access control framework*, ISO/IEC.
- [20] FIDIS Online: <http://www.fidis.net/>
- [21] Sebastian Clauí, Dogan Kesdogan, Tobias Klopsch, Lexi Pimenidis, Stefan Schiffner, Sandra Steinbrecher: *Privacy Enhancing Identity Management: Protection Against Re-identification and Profiling*. ACM CCS2005 Workshop on Digital Identity Management, November 2005, George Mason University, Fairfax, VA, USA.
- [22] W. A. Jansen, "Contermeasures for Mobile Agent Security", *Elsevier Computer Communications*, vol. 23, Issue. 17, November 2000, pp. 1667-1676.
- [23] L. Gong and R. Schemers, "Signing, Sealing and Guarding Java Objects", *Lecture Notes in Computer Science*, vol. 1419, Springer-Verlag, 1998, pp. 206-216.
- [24] G. Karjoth D. B. Lange and M. Oshima, "A Security Model for Aglets", *IEEE Internet Computing*, vol. 1 No. 4, July 1997, pp. 68-77.

-
- [25] S. Berkovits, J. D. Guttman and V. Swarup, "Authentication for Mobile Agents", Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, 1998, pp. 114-136.
- [26] W. A. Jansen, "A Privilege Management Scheme for Mobile Agent Systems", in Proc. of the 1st International Workshop on Security of Mobile Multi-agent Systems, Autonomous Agents Conference, Montreal, Canada, May 2001.
- [27] R. Wahbe, S. Lucco, T. E. Anderson, S. L. Graham, "Efficient Software-based Fault Isolation", ACM SIGOPS Operating Systems Review, vol. 27 No. 5, December 1993, pp. 203-216.
- [28] W.M. Farmer, J.D. Guttman and V. Swarup, "Security for Mobile Agent: Authentication and State Appraisal", Lecture Notes in Computer Science, vol. 1146, Springer-Verlag, 1996, pp. 118-130.
- [29] G. C. Necula, "Proof-carrying code", in Proc. of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Paris, France, 1997.
- [30] D. Chess, B. Grosz, C. Harrison, D. Levine, C. Parris and G. Tsudik, "Itinerant Agents for Mobile Computing", IEEE Personal Communications, vol. 2, No. 5, 1995.
- [31] C. Cao and J. Lu, "A Path-history-sensitive Access Control Model for Mobile Agent Environment", in Proc of the 3rd International Workshop on Mobile Distributed Computing (IEEE ICDCSW'05), Washington DC, USA, 2005.
- [32] B. S. Yee, "A sanctuary for mobile agents", Technical Report CS97-537, UC San Diego, Department of Computer Science and Engineering, April 1997.
- [33] R. R. Jueneman, S. M. Matyas and C. H. Meyer, "Message authentication", IEEE Communications Magazine, vol. 23, No. 9, 1985, pp. 29-40.
- [34] G. Vigna, "Cryptographic Traces for Mobile Agents", Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, 1998, pp. 137-153.
- [35] Y. Chen, R. Venkatesan, M. Cary, R. Pang, S. Sinha and M. H. Jakubowski, "Oblivious Hashing: A Stealthy Software Integrity Verification Primitive", Lecture Notes in Computer Science, vol. 2578, Springer-Verlag, 2002, pp. 400-414.
- [36] J. Riordan and B. Schneier, "Environmental Key Generation Towards Clueless Agents", Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, 1998, pp. 15-24.
- [37] F. Hohl, "Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts", Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, 1998, pp. 92-113.
- [38] T. Sander and C. F. Tschudin, "Towards Mobile Cryptography", in Proc. of the 1998 IEEE Symposium on Security and Privacy, Oakland, USA, May 1998.
- [39] Web Services Security: SOAP Message Security 1.1, 1 February 2006, available at <http://www.oasis-open.org/committees/wss/>
- [40] W3C Recommendation, "XML Encryption Syntax and Processing", 10 December 2002, available at <http://www.w3.org/TR/xmlenc-core/>
- [41] W3C Recommendation, "XML-Signature Syntax and Processing" 12 February 2002, available at <http://www.w3.org/TR/xmldsig-core/>
- [42] Web Service Trust Language, February 2005, <http://www-128.ibm.com/developerworks/webservices/library/specification/ws-trust/>
- [43] XACML profile for Web Services, 29 September 2003, available at <http://www.oasis-open.org/committees/download.php/3661/draft-xacml-wspl-04.pdf>
- [44] Web Service Policy Framework, Version 1.2, March 2006, <http://www-128.ibm.com/developerworks/library/specification/ws-polfram/>
- [45] Web Service Policy Attachment, Version 1.2, March 2006, <http://www-128.ibm.com/developerworks/webservices/library/specification/ws-polatt/>

-
- [46] Web Service Policy Assertions, Version 1.2, March 2006, <http://www-128.ibm.com/developerworks/webservices/library/specification/ws-polas/>
- [47] OMG CORBA Secure Service Specification, Version 1.8, March 2002, available at <http://www.omg.org>
- [48] N. Li and J.C. Mitchell, "Securing Java RMI-based Distributed Applications", in Proc. of Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04), December 2004.
- [49] D. Zalewski, "Security Enhancement of Java Remote Method Invocation", in Proc. of the International Conference on Dependability of Computer Systems, Wroclaw, May 2006.
- [50] N. Santos, P. Marques and L. Silva, "A Framework for Smart Proxies and Interceptors in RMI", in Proc. of the First International Mobile IPRWorkshop: Rights Management of Information (MobileIPR 2003), Helsinki-Finland, August 2003.
- [51] Ch. Wang, A. Carzaniga, D. Evans and A.L. Wolf, "Security Issues and Requirements for Internet-Scale Publish-Subscribe Systems", in 35th Hawaii International Conference on System Sciences, Big Island, Hawaii, January 2002.
- [52] A. Carzaniga, D.S. Rosenblum and A.L. Wolf, "Design and evaluation of a widearea event notification service", ACM Transactions on Computer Science (TCS), vol. 19, no. 3, August 2001, pp. 332-383.
- [53] M. Klonowski, M. Kutylowski, B. Rozanski, "Privacy Protection for P2P Publish-Subscribe Networks", Security and Protection of Information 2005, Brno University of Defence 2005, ISBN 8085960-99-0, pp. 63-74.
- [54] Z. Miklos, "Towards an access control mechanism for widearea publish/subscribe systems", In Proc. of the International Workshop on Distributed Event-based Systems (DEBS'02), Vienna, Austria, July 2002.
- [55] S. Pallickara, M. Pierce, G. Fox, Y. Yan and Y. Huang, "A Security Framework for Distributed Brokering Systems", http://grids.ucs.indiana.edu/ptliupages/publications/NBSecurityFramework_acmcss.pdf
- [56] A. Belokosztolszki, D. M. Eyers, P. R. Pietzuch, J. Bacon and K. Moody, "Rolebased access control for publish/subscribe middleware architectures", in Proc. Of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03), San Diego, California, June 2003.
- [57] J. Bacon, K. Moody and W. Yao, "A model of OASIS role-based access control and its support for active security", ACM Transactions on Information and System Security (TISSEC), vol. 5, no. 4, November 2002, pp. 492-540.
- [58] P.R. Pietzuch and J.M. Bacon, "Hermes: A Distributed Event-Based Middleware Architecture", in Proc. of the 1st International Workshop on Distributed Event-Based Systems (DEBS'02), Vienna, Austria, July 2002.
- [59] L. Fiege, A. Zeidler, A.P. Buchmann, R. Kilian-Kehr and G. Mühl, "Security aspects in publish/subscribe systems", in Proc. of the Third Intl. Workshop on Distributed Event-based Systems (DEBS'04), Edinburgh, Scotland, UK, May 2004.
- [60] G. Muhl, "Large-Scale Content-Based Publish/Subscribe Systems", PhD thesis, Darmstadt University of Technology, 2002. <http://elib.tudarmstadt.de/diss/000274/>
- [61] Ants, <http://antsp2p.sourceforge.net/>
- [62] Freenet, <http://freenetproject.org/>
- [63] Entropy, <http://entropy.stop1984.com/en/home.html>
- [64] GUNet, <http://gnunet.org/>
- [65] I2P, <http://www.i2p.net/>
- [66] MUTE, <http://mute-net.sourceforge.net/>
- [67] Nodezilla, <http://www.nodezilla.net/>
- [68] Rodi, <http://rodi.sourceforge.net/>

-
- [69] Share, <http://www.uguu.org/share/>
- [70] Sumi, http://sumi.berlios.de/wiki/Main_Page
- [71] TOR, <http://tor.eff.org/>
- [72] DC++, <http://www.dcpp.net/>
- [73] Waste, <http://waste.sourceforge.net/>
- [74] M. Blaze, J. Feigenbaum, J. Ioannidis and A.D. Keromytis, "The Role of Trust Management in Distributed Systems Security", In Secure Internet Programming: Issues in Distributed and Mobile Object Systems, Springer-Verlag Lecture Notes in Computer Science State-of-the-Art series, 1999, pp. 185-210.
- [75] G. Zacharia and P. Maes, "Trust Management Through Reputation Mechanisms", Applied Artificial Intelligence, vol. 14, 2000, pp. 881-907.
- [76] F. Cornelli, E. Damiani, S.C. Vimercati, S. Paraboschi, and P. Samarati, "Choosing reputable servants in a P2P network", In Proc. of the Eleventh International Conference on World Wide Web, Honolulu, Hawaii, USA, 2002.
- [77] S. Lee and R. Sherwood, "Cooperative peer groups in NICE", IEEE Infocom 2003, San Francisco, USA.
- [78] M. Gupta and P. Judge, "A Reputation System for Peer-to-Peer Networks", in Proc. of the Thirteenth ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video, Monterey, California, 2003.
- [79] F. Cornelli, E. Damiani, S.C. Vimercati, S. Paraboschi and P. Samarati, "A reputation-based approach for choosing reliable resources in peer-to-peer networks", In Proc. of CCS'02, Washington DC, USA, 2002.
- [80] S. Kamvar, M. Schlosser and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks", in Proc. of the Twelfth International World Wide Web Conference, Budapest, Hungary, May 2003.
- [81] J. Sabater and C. Sierra, "Reputation and social network analysis in multi-agent systems", in Proc. of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy, 2002.
- [82] J. Pujol and R. Sanguesa, "Extracting reputation in multi agent systems by means of social network topology", in Proc. of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy, 2002.
- [83] P. Bonatti, Cl. Duma, D. Olmedilla and N. Shahmehri, "An Integration of Reputation-based and Policy-based Trust Management", in Proc. of Semantic Web Policy Workshop (SWPW 05), Galway, Ireland, November 7, 2005.
- [84] David Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms", Communications of the ACM 4(2), February 1981 .
- [85] "Mixmaster and remailer attacks" L. Cottrell, <http://www.obscura.com/.loki/remailer/remailer-essay.html>.
- [86] Roger Dingledine, Nick Mathewson, and Paul Syverson "Tor: The Second- Generation Onion Router", In the proceedings of the 13th USENIX Security Symposium, August 2004.
- [87] "Hiding Routing Information", Information Hiding, R. Anderson (editor), Springer-Verlag LNCS 1174, 1996, pp. 137-150.
- [88] <http://www.onion-router.net/>
- [89] The TLS Protocol. Version 1.0. IETF RFC 2246 T. Dierks and C. Allen. January 1999. <http://www.rfc-editor.org/rfc/rfc2246.txt>.

- [90] George Danezis, Roger Dingledine and Nick Mathewson "Mixminion: Design of a Type III Anonymous Remailer Protocol" In the Proceedings of the 2003 IEEE Symposium on Security and Privacy, May 2003.
- [91] <http://www.iusmentis.com/technology/remailers/index-penet.html>
- [92] <http://www.firstmonday.org/issues/issue2/remailers/index.html>
- [93] "Mixmaster Protocol . Version 2." U. Moller and L. Cottrell Unfinished draft, January 2000. <http://www.eskimo.com/~rowdenw/crypt/Mix/draft-moeller-mixmaster2-protocol-00.txt>.
- [94] Michael J. Freedman and Robert Morris. "Tarzan: A Peer-to-Peer Anonymizing Network Layer". In the Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), Washington, DC, November 2002.
- [95] M. Gruteser, D. Grunwald – "Enhancing Location Privacy in Wireless LAN Through Disposable Interface Identifiers: A Quantitative Analysis".
- [96] L. Huang, K. Matsuura, H. Yamane, K. Sezaki – "Enhancing Wireless Location Privacy Using Silent Period".
- [97] Y. Hu and H. Wang - "A framework for Location Privacy in Wireless Networks".
- [98] M. Gruteser, D. Grunwald – "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking".
- [99] A. Pfitzmann. And M. Koehntopp, "Anonymity, unobservability, and pseudonymity– a proposal for terminology".
- [100] G. Borriello and L. Holmquist, "A Privacy Awareness System for Ubiquitous Computing Environments".
- [101] K. Sampigethaya, L. Huang, M. Li, R. Poovendran, K. Matsuura, K. Sezaki, "CARAVAN: Providing Location Privacy for VANET".
- [102] W. Du, J. Deng, Y.S. Han, S. Chen, P.K. Varshney, "A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge".
- [103] C. Karlof, D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures".
- [104] A. Juels, R.L. Rivest, M. Szydlo: Selective blocking of RFID tags for consumer privacy. In 10th Annual ACM CCS 2003, May 2003.
- [105] S. E. Sarma, S. A. Weis, and D.W. Engels. Radio-frequency identification systems. In Burton S. Kaliski Jr., C. etin Kaya Ko,c, and Christof Paar, editors, CHES '02, pp. 454-469. Springer-Verlag, 2002. LNCS no. 2523.
- [106] <http://www.mobilecloak.com>
- [107] K.P. Fishkin, S. Roy, Enhancing RFID Privacy via antenna Energy Analysis, MIT RFID Privacy Workshop, Boston, November 2003.
- [108] S. A. Weis, S. Sarma, R. Rivest, and D. Engels. Security and privacy aspects of lowcost radio frequency identification systems. In First International Conference on Security in Pervasive Computing, 2003.
- [109] A. Juels and R. Pappu. Squealing Euros: Privacy protection in RFID-enabled banknotes. In R. Wright, editor, Financial Cryptography '03. Springer-Verlag, 2003.
- [110] A. Juels, R. L. Rivest, and M. Szydlo, The blocker tag: Selective blocking of RFID tags for consumer privacy in Proc. 8th ACM Conf. Comput. Commun. Security, V. Atluri, Ed., 2003, pp. 103-111.
- [111] C. Floerkemeier, R. Schneider, and M. Langheinrich. (2004) Scanning with a purpose—Supporting the fair information principles in RFID protocols. [Online]. Available: citeseer.ist.psu.edu/floerkemeier04scanning.html
- [112] A. Khalili, W. A. Arbaugh, "Security of Wireless Ad Hoc Networks", 2002.

-
- [113] D. Harkins, D. Carrel, "The Internet Key Exchange (IKE)", IET RFC 2409, November 1998.
 - [114] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", November 1998.
 - [115] H. Orman, "The OAKLEY Key Determination Protocol", IETF RFC2412, November 1998.
 - [116] H. Krawczyk, "SKEME: A Versatile Secure Key Exchange Mechanism For Internet", IEEE, 1996.
 - [117] D. Maughan, M. Schertler, J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", IETF RFC 2408, November 1998.
 - [118] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol", IETF RFC4306, December 2005.
 - [119] N. Asokan and P. Ginzboorg, "Key-Agreement in Ad Hoc Networks", Computer Communications, vol. 23, no. 17, pp. 1627-1637, 2000.
 - [120] A. Shamir, "How to Share a Secret", Communications of the ACM, vol.22, pp. 612-613, November 1979.
 - [121] Y. Desmedt, "Some Recent Research Aspects of Threshold Cryptography", Proc. Information Security, (Lecture Notes in Computer Science 1396), pp. 158-173, Springer-Verlag, 1997.
 - [122] L. Zhou, Z. Haas, "Securing ad hoc networks", IEEE Network Magazine, 1999.
 - [123] S. Yi and R. Kravets. "MOCA: Mobile certificate authority for wireless ad hoc networks." In Proceedings of the 2nd Annual PKI Research Workshop (PKI 03), Apr. 2003.
 - [124] H. Luo, S. Lu, "URSA: ubiquitous and robust access control for mobile ad-hoc networks", IEEE/ACM Trans Networking, 2004.
 - [125] J-P Hubaux, L. Buttyan, and S. Capkun, "The Quest for Security in Mobile Ad hoc Networks", in proc. MobiHoc, Lausanne, Switzerland, 2002.
 - [126] S. Capkun, L. Buttyan et J-P Hubaux, "Self-Organized Public Key Management for Mobile Ad Hoc Networks", ACM International Workshop on Wireless Security, (WiSe 2002), 2002.
 - [127] S. Capkun, L. Buttyan, J. Hubaux, "Self-organized public-key management for mobile ad hoc networks", IEEE Trans Mobile Comput, 2003.
 - [128] R. Struik and G. Rasor, "Mandatory ECC Security Algorithm Suite", submissions to IEEE P802.15 Wireless Personal Area Networks, March 2002.
 - [129] M. Aydos, T. Yan and C. K. Koc, "A High-speed ECC-based Wireless Authentication Protocol on an ARM Microprocessor", 16th Annual Computer Security Applications Conference (ACSAC'00), December 2000.
 - [130] SECG, Elliptic Curve Cryptography, Standards for Efficient Cryptography Group, 2000.

11 Conclusions

The advancement expected from the research and development activities undertaken in the HYDRA project in the technology areas described in previous sections, is outlined below.

11.1 Embedded, autonomous AmI Architecture

Services for ambient Intelligence

HYDRA plans to offer embedded Ambient Intelligence capabilities through an appropriate architecture providing semantic support. Once a device is plugged into the HYDRA middleware, a knowledge model describing the capabilities of each device is available. The ultimate goal of the devices is to serve the users, and to this end, the users need to specify what they want to do. Due to the flexibility of the knowledge (domain) models future extensions into e.g. perception and cognition may be facilitated.

Embedded and mobile service-oriented architectures for AmI

Service-Oriented Architecture may be thought of as a novel way to structure computing, not just a new mean of integration for existing applications. In a world of ambient intelligence, services typically reside on small, embedded devices that feature certain communication capabilities and are spread to cover wide areas with the services offered. The specific challenges of HYDRA then pertain to making such architectures sufficiently embeddable and mobile. Given the complex, changing, almost chaotic environment, one of the HYDRA objectives is to build a network through which all devices can discover each other and communicate.

Autonomous Computing

When working with devices, an important issue is the discovery of the devices that could fall inside the scope of the controllers. The emergence of information appliances and new types of connectivity are inciting a new form of networking: dynamic networks of consumer devices that join and leave the network. At the moment some discovery protocols that are relevant to HYDRA are available, but the protocols should be interoperable so that any device supporting a standard protocol can communicate with other devices. Good networking solutions should be able to accommodate heterogeneity, both in terms of underlying connectivity, and in terms of the discovery infrastructure that is supported.

11.2 Wireless Networks and devices

Grid service based architecture

A web-based sensor network would make all repositories of sensor data immediately discoverable, accessible and controllable. If ontology-based models of the sensors are available, the sensors can be accessed through metadata requests.

Another way of achieving the same result is by using Grid technologies, which is one of the approaches used in HYDRA. Grid applications are data-centric and focused on distributed services. Much of the data for these applications are collected by a new class of small, self-contained, intelligent devices that combine limited sensing and computation capabilities with wireless communications.

Communication mechanisms

The ubiquitous nature of the embedded systems affects also types of technologies used for communication between devices. Different wireless technologies need to be supported by the HYDRA middleware. A high level classification could distinguish protocols in relation to their operation area.

Technologies like ZigBee, Bluetooth, Wi-Fi are going to be integrated in the HYDRA middleware, so that all PAN, LAN, and WAN protocols will be transparent to system-device interaction. All commands, data flow and functionality finding will be processed by this middleware, so system will communicate with this middleware instead of with each individual protocol.

Wireless networks

There are several forms of wireless connectivity. The HYDRA middleware will be capable of handling devices using various protocols in various combinations. The emerging domain of wireless sensor networks has produced lots of different devices with very different computation, communication, and sensing capabilities. The fundamental objective for wireless device networks are reliability, accuracy, flexibility, cost effectiveness and ease of deployment. HYDRA will use derived grid technologies in order to implement an efficient decentralised device network. This will facilitate the use of open standards, general-purpose protocols and interfaces as well as authentication, authorization, resource discovery and access.

11.3 SoA and MDA middleware

Embedded Service-Oriented Architecture for AmI applications

Service-Oriented Architecture

Service Oriented Architecture (SoA) represents an architectural style where the primary concept is the use of loosely coupled, implementation-neutral services supporting a business process as building blocks. Service consumers use the service by means of its published interface-based service description without dependence on implementation, location or technology.

All of the software components comprising HYDRA will be integrated in a *Service Oriented Architecture*, which will provide, among other things, interoperability. The HYDRA middleware will thus become the link between web services and devices. Interoperability, which here is taken as the capability of components of HYDRA to talk to each other no matter which is the technology used to implement them or their physical location, is achieved by means of the usage of many specifications around the web services world. The main purpose of the Service-oriented Architecture in Hydra is to provide interoperability between devices.

An Embedded Mobile Computing platform

The HYDRA middleware will support applications with fixed and mobile artefacts and terminals, through which the end-users will access the HYDRA functionalities, which can range from a desktop browser to a mobile phone, and from having only presentation requirements to also need of local data and processing capabilities, and thus of synchronization of data.

The HYDRA middleware will thus comprise a comprehensive and coherent Mobile Computing platform that will ease the task of creating applications for mobile devices and terminals.

Semantic Model Driven Architecture for AmI applications

Semantic Web

Web technologies are shifting towards providing an automated environment for delivering a wide variety of business-to-consumer and business-to-business services and applications such as the ones envisioned in HYDRA. Such services and applications will communicate and interoperate in a world composed of Web accessible programs and databases, and interface wirelessly with many smart devices and sensors. These shifts have the potential to change significantly the way we communicate, cooperate, and organise our commercial and personal relationships.

The Semantic Web is fundamental to enabling these types of services and applications by providing a universally accessible platform that allows data to be shared and processed by automated tools, and by providing the machine-understandable semantics of data and information that will enable automatic information processing and exchange.

Model-Driven Architecture

The model-driven architecture (MDA) process places formal system models at the core of the interoperability problem. What is most significant about this approach in relation to HYDRA is the independence of the system specification from the implementation technology or platform. The system definition exists independently of any implementation model and has formal mappings to many possible platform infrastructures.

In order to cope with the lack of standardization of one middleware, HYDRA will design a high-level layer that is compliant with the MDA. Thus, HYDRA will get portability, interoperability, and reusability as well as reducing cost of business whenever developers have to make software

migration. A specific challenge will be to handle semantic techniques and technologies in a model-driven platform.

Domain Modelling: A high-level semantic approach

HYDRA aims to interconnect devices, people, terminals, buildings, etc. The Service-Oriented Architecture and its related standards provide interoperability at a syntactic level. However, in HYDRA we also aim at providing interoperability at a *semantic level*. Thus, the HYDRA middleware must also contain domain models modelling particular parts of the world. The way to achieve a flexible decoupling between higher levels describing domain terms, relations and semantics and the underlying syntactic operative infrastructure is based on the adoption of Semantic Web Technologies.

Automatic Device Classification and Ontology Design

In order to cope with the huge variety of capabilities of the devices to be integrated in HYDRA, it is important to be able to describe the capabilities of the devices in such way that an automatic agent can understand these capabilities and use them (i.e. to use ontology-based models). Once the semantics describing the model of the other device has been found, then the device capabilities could be accessed.

11.4 Trust, privacy and security

Trust, privacy and security

In a world where physical boundaries are eroding rapidly, security depends on building-in and being able to manage the logical boundaries appropriately. Existing solutions, however, are not integrated in a general security system: the diversity of techniques and methods has increased risks and vulnerabilities and require integrated processes. Another challenge to be overcome is the lack of the standards definitions available for virtualisation regarding security as well as for the integration of different component and protocols and context-aware security model invocation.

HYDRA aims to enhance the resolution of this gap by providing a visible and controllable technology, which is based on the concept of trust as a multilateral relation between end users in a community of other users, actors and stakeholders. Trust requires adapting policies dynamically and domain-dependently to changing requirements and could be transferred like patterns from one domain to the other.

In HYDRA both the security and trust chain as well as privacy concerns will be identified and ranked in terms of their priority so that the virtualisation, secure transaction models and secure Identity Management can be accommodated effectively and efficiently everywhere.

Inter-operable Security Engineering

The integration of different technologies addressing the various requirements of interconnected clients has to be based on a flexible modular infrastructure supporting the existing legacy technologies as well as being capable of integration with new technologies as they become available. To support the establishment of a generic modular integrated security infrastructure an iterative process of security metrics and models requirements engineering, design specification and implementation has to be undertaken towards a robust solution. Such a solution will be expected to deploy model-driven and service-oriented layers as well as semantic cooperative standards and generalisation ontologies and be able to accommodate various security contexts such as centralised distributed security and service management architectures.

Distributed Identity Management

With the increase of cross-boundary services that span over several organizations or web sites and involve multiple agents, the need for a distributed way to manage the user identity becomes an increasingly concerning issue. This problem is closely related with single-sign on solutions, and a strong desire to reduce the amount of login operations that users must undergo. Currently several solutions exist that provide a solution for this problem, but none of them is fully satisfactory. A lot of other distributed identity systems are not actually distributed, having one or more parts centrally controlled.

In order to implement an application middleware for model based applications, keeping privacy and audit trails so as to guarantee users their safety when performing transactions with the middleware and to ease the adherences to different regional legislations for electronic transactions it is mandatory to have some identity management system.

Distributed Trust

HYDRA will use a distributed federation of trust. Elements in the HYDRA network will trust ones in others to the proper degree in each case (i.e. depending on actual authorizations), and when appropriate this trust could be transferred to other elements (e.g. if A trusts B and B trusts C, then A can trust C). Ultimately, all trusts can be traced back to the owner of a device (who initiated the chain of trust).

Security and access control

Integration of a security system that will grant or deny access to the created contents based on the user identity will be considered in the HYDRA middleware. By their very nature, multi-user distributed networked systems are insecure in the sense that exchanged data is exposed to eavesdropping and modifications done on its way through the network. Many of the protocols used in these kind of systems do not provide any kind of security, and plenty applications rely on the other end of the communications to be "honest" about his identity.

Some approach must be taken to restrict access to the middleware based on user/company defined policies and to classify services according to their security restriction. For web service composition, possible solution is integration of Web Services Security (WS-Security) in the platform, which is designed to help organizations build secure, broadly interoperable Web service applications by accommodating various security models as well as digital signatures and encryption technologies, and has already working open-source implementations.

Virtual Identities and Authentication

An Identity Security infrastructure (with minimum overhead) is required to block actions from identities. In HYDRA we distinguish between various types of identities such as virtual identities, federated identities with respect to different domains. Identity virtualisation has to be combined with infrastructure-based accountability negotiations, which are very efficient but also complex. HYDRA will assess the feasibility of approaches, which incorporates a context server in the server-side structure taking care of next generation session initiation. This can be combined with a client device layer that will take care of device and identity virtualisation and security adaptation to context. In addition identity management will need a way to transfer between trust domains.