

IOP principles		1 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09



Smart Objects For Intelligent Applications

D5.11

Interoperability Platform Principles

Legal disclaimer

The material contained herein is Confidential Information which may only be used in accordance with the terms of the SOFIA Project Consortium Agreement (PCA).

Access Rights needed for the execution of the SOFIA project are granted unless excluded in Annex IV of the PCA. No other licenses to any related IPR are implied.

The material is protected by copyright laws, and may not be reproduced, distributed or otherwise exploited in any manner without the prior permission of the rights holders.

The contributors are not liable for the use of this material except as stated in the SOFIA PCA.

Change History

Version	Date	Description	Affected Sections
0.01	2009-02-23	Draft version of SOTA	1
0.01	2009-03-06	Draft version of Requirements	2
0.02	2009-03-17	VTT: Requirements collected from wp1 and wp3 scenarios	2.1
0.03	2009-03-23	VTT: Security requirements	Security (2.1)
0.04	2009-03-23	UNBO: Requirements from WP2 scenarios	2.1
0.05	2009-03-26	UNBO: Refinement and merging of requirements from all scenarios; draft proposal for summary and conclusions	2
0.06	2009-03-29	UNRO: revision of the whole document; improvements in the inclusion of wp3 scenarios	2
0.07	2009-06-10	VTT: Refined prioritization of requirements; minor refinements to other sections	2
0.08	2009-06-10	UNBO: minor refinements to prioritization section	2.2
0.09	2009-06-16	UNBO: inclusion of principles and criteria	Added section
0.10	2009-06-18	UNBO: first amalgamation	ALL
0.11	2009-06-21	UNBO: refined after partner comment	3
0.12	2009-06-23	UNBO: refined after partner comment	3
0.13	2009-09-09	UNBO and VTT: design of KP template based on a previous input from WP3 (ED et al.)	5
0.14	2009-10-25	Updated KP template with an example of an implemented KP	5
0.15	2009-12-10	Added the <i>productivity principle</i> to IOP principles list Added Smart Environment deployment strategy Added guidelines to design a smart space	3, 4
0.16	2009-12-27	Added KP Taxonomy and KP versus Ontology Table Added Guidelines to further research on KPs	4
1.0	2010-01-08	Merge of sections 4 and 5 with sections [1..3]	1, 4, 5

List of Contributors

Participating Entity	Contributing Individuals
VTT	Eila Ovaska, Susanna Pantsar-Syväniemi, Antti Evesti, Katja Henttonen, Daniele Manzaroli (Daniele is from UNBO, but he was at VTT when he contributed)
UNBO	Paolo Bellavista, Alessandra Toninelli, Rebecca Montanari, Carlo Giannelli, Alfredo D'Elia, Guido Zamagni, Fabio Vergari, Sara Bartolini, Luca Roffia, Federico Spadini, Tullio Salmon Cinotti
UNIROMA	Roberto Baldoni, Sirio Scipioni, Leonardo Querzoni
ELSAG DATAMAT	Paolo Pucci, Giorgio Laura

Table of Contents

EXECUTIVE SUMMARY	5
1. STATE-OF-THE-ART OF SMART ENVIRONMENTS TECHNOLOGIES	7
1.1. STATE-OF-THE-ART OF SMART ENVIRONMENTS AND CONTEXT MANAGEMENT	7
1.1.1. Context, Context-awareness, Context-aware Techniques.....	8
1.1.2. Context-aware architecture.....	9
1.1.3. Metadata-Based Context Representation.....	11
1.1.4. Context Management and Provisioning in Mobile Environments.....	12
1.1.5. Context Management Middleware Infrastructures.....	15
1.1.6. Ontology-based Approaches to Context Representation and Reasoning	17
1.1.7. Existing Context Ontologies for SOFIA Application Domains	20
1.1.8. Event-based communications.....	21
1.1.9. References.....	23
1.2. STATE-OF-THE-ART OF ONTOLOGY-ORIENTED SECURITY MANAGEMENT	26
1.2.1. Security Ontologies.....	26
1.2.2. QoS Ontologies.....	34
1.2.3. ONTOMETRIC	35
1.3. OVERVIEW OF SECURITY DESIGN	36
1.3.1. Defining Security at Design-time.....	36
1.3.2. Defining Security at Run-time.....	37
1.3.3. References.....	39
1.4. ANNEX 1: RAPID OVERVIEW OF POLICIES FOR NETWORKS, SYSTEMS AND SERVICE MANAGEMENT	41
1.4.1. References.....	42
2. IOP REQUIREMENTS FROM VERTICAL SCENARIOS	43
2.1. REQUIREMENTS	44
2.1.1. Quality Requirements.....	44
2.1.2. Functional Requirements	60
2.2. PRIORITIZATION OF THE IOP REQUIREMENTS.....	70
3. IOP PRINCIPLES AND CRITERIA TO EVALUATE THE IOP.....	73
3.1. IOP PRINCIPLES	73
3.1.1. Introduction.....	73
3.1.2. Principles.....	74
3.2. CRITERIA TO EVALUATE THE IOP.....	77
3.2.1. Motivation.....	77
3.2.2. Criteria for Functional Evaluation.....	78
3.2.3. Criteria for Development Time Quality Properties Evaluation	78
3.2.4. Criteria For Execution Time Quality Properties Evaluation	79
4. GUIDELINES TO DESIGN A SMART SPACE.....	80
4.1. SMART ENVIRONMENTS DEPLOYMENT STRATEGY	80
4.1.1. Introduction.....	80
4.1.2. Terminology	80
4.1.3. KP development approaches.....	81
4.1.4. KP taxonomy.....	83
4.1.5. Guidelines to further research on KPs.....	84
4.1.6. Smart Environment design steps	86
4.2. DESCRIBING A SMART ENVIRONMENT	88
4.2.1. Hints for a “Smart Environment Template”	88
4.2.2. Motivation and structure of the “KP Template”	89
4.3. REFERENCES.....	89
5. TEMPLATES	90
5.1. IOP REQUIREMENTS TEMPLATE	90
5.2. TEMPLATE PROPOSAL FOR APPLICATION SPECIFIC AND ADAPTABLE KPS	91



IOP principles		4 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

APPENDIX: IOP REQUIREMENTS QUICK REFERENCE GUIDE 98

IOP principles		5 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Executive Summary

D5.11 is about "IOP principles" and it includes the following sections:

1. State-of-the-art on Smart Environments Technologies
2. IOP Requirements from Vertical Scenarios
3. IOP Principles and Criteria to Evaluate the IOP
4. Guidelines to Design a Smart Space
5. Templates

In section 1 the state-of-the-art on most relevant smart environment technologies is presented, including ontology based context and security management.

In Section 2 the IOP requirements are distilled from the scenarios defined for personal spaces (WP1), smart indoor spaces (WP2) and smart cities (WP3), and are then prioritized based on the assessment criteria defined by the Project Management Team.

The motivation behind the IOP requirements specification from the analysis of the vertical scenarios is to derive appropriate principles for *SOFIA InterOperability Platform (IOP)*. *SOFIA IOP* is an infrastructure to assist its users with added-value interoperable information about objects existing in the user's environment. It combines *SOFIA* information interoperability solutions with existing/legacy solutions for service and physical level interoperability.

IOP speciality is its information interoperability level. The interoperability is based on common ontology models used in modelling information. Information may originate from heterogeneous legacy and embedded devices spread in the environment or may be produced by aggregators of IOP information. *IOP* information repository may be active in the sense that it may notify external entities, thus triggering their reaction to relevant and selected events.

The following IOP "qualities" are considered most important:

1. its ability to generate business
2. its application engineering process (i.e., design, development, and deployment of the applications)
3. the interaction models (interaction among users, their environment, and their history)
4. security and dependability (i.e., *IOP* should respect privacy, enforce access policy rules, and ensure information integrity and trust)
5. performance, energy efficiency, and scalability

An *IOP* with these qualities is expected to start a market for emerging applications that can interoperate independently from their business/vendor/manufacturer origin, introducing in this way a radical change to the traditional application scenario which is based on fixed business boundaries. Expected applications may be cross-domain, may adapt to the user situation, may spontaneously start when required and have the potential for significant market penetration and socio-economic impact.



IOP principles		6 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

With the above vision in mind, 16 principles are adopted by SOFIA and they should be reflected in the *IOP* Big Picture, in the *IOP* Reference Logical Architecture, and in any *IOP* implementation.

Section 3 states and motivates the 16 *IOP* principles and proposes a set of criteria to evaluate *IOP* implementations, aiming to verify if such *IOP* instances (localizations) satisfy the requirements identified by the vertical Workpackages. It is expected that if an *IOP* localization meets the 16 principles, then not only the *IOP* requirements distilled from the vertical scenarios, but also the ones set by many future high-impact smart environment applications will be satisfied. This hypothesis should be verified by T5.4.

Section 4 is the result of the interaction between verticals and horizontal WPs during the first 11 months of the project. It is inspired by the design experience gained by SOFIA partners in 2009 and it has the ambition to provide guidelines on how to design a Smart Space.

It is addressed to the vertical WP partners and to candidate Smart Environment “Governors” who have management rights over a specific real space and intend to turn such space into a Smart Environment. It envisages a smart environment design and deployment strategy, it compares two implementation approaches, and it suggests some smart space-related topics deserving further research.

Section 4 includes the initial version of the KP taxonomy. The KP taxonomy is expected to evolve when more experience on smart space development and deployment is achieved. The aim of the KP taxonomy is to classify the existing KPs into a set of clusters that help smart space developers to find out suitable KPs for their development work. Thus, the KP taxonomy will help in smart space evolution management and will also evolve itself. Furthermore, Section 4 lists some criteria to describe and specify a smart space, and suggests a number of relevant templates, some of which are collected at the end of the deliverable in Section 5. These templates are also expected to evolve and be refined during the second and third project years.

For all the motivations listed above, we claim that T5.1 activities should not terminate with this deliverable, at project time t_{12} as anticipated in SOFIA TA. T5.1 partners suggest continuing templates and taxonomy refinement in the next months. If this is done, then we may expect that a comprehensive, optimized, validated, and mature smart environment deployment strategy is available by the end of the project: this would be one of the main results of the interaction between horizontal and vertical WPs.

1. State-of-the-art of Smart Environments Technologies

In this section we provide the state-of-the-art on some of the smart environment technologies that should be of interest for the SOFIA project (WP5).

In particular, related work is organized in the following three subsections, regarding:

1. General description of smart environments, context awareness and related models; this subsection includes:
 - architectural and representational models to support context provisioning and management, especially in mobile embedded systems;
 - ontology-based approaches to context awareness, including existing context ontologies – with a special focus on those that could be exploited in SOFIA.
2. State of the art of ontology oriented security management; this subsection includes:
 - security and Quality of Service (QoS) ontologies
 - the ONTOMETRIC method that facilitates ontology selection.
3. Security management at design time and at run time.

An annex with a rapid overview of policies for networks, systems and service management is also included.

Each of the above subsections and annex is concluded with its own reference list.

1.1. State-of-the-art of Smart Environments and Context Management

A major target in SOFIA-project is to create an interoperable platform (IOP) to enable different kind of objects to interact. The objects can be e.g. sensors, devices, appliances, and embedded systems. These interacting objects form different kind of smart environments. IOP will be the platform for smart applications, i.e. to enable supply of new and innovative services for smart environments. In other words, we aim to connect physical world with information world by enabling the sharing of information in digital format in physical spaces. This, sharing of information, makes possible to create novel and possibly cross-domain applications. In Figure 1 is illustrated a couple of objects who are sharing the information between each other. The objects share only that information they want other object to see or use in the situation. The shared information is dependent on the context.

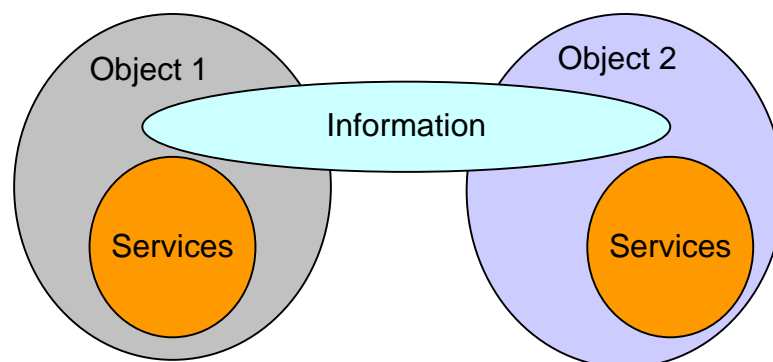


Figure 1. Objects sharing information between each other.

IOP principles		8 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

In [1] is introduced a hierarchical, agent-based service architecture for smart environments. That architecture is called Smart Environment Hierarchy (SETH). It allows creating complex smart environments by combining a certain number of smaller smart places like rooms to create bigger smart places like buildings. In Figure 2 is shown one example of hierarchical arrangements of smart spaces.

In [1] ‘smart space’ is used as a synonym for ‘smart environment’. SETH architecture relies on the concept of smart spaces which can be indoor or outdoor spaces, and they may be also static or mobile, e.g. a smart car. In SETH a smart space is characterized by a set of *devices*, a set of available *services*, and a given *context*.

SOFIA recognizes as well as SETH, the three key issues for smart environment: interacting objects (devices in SETH), available services, and context.

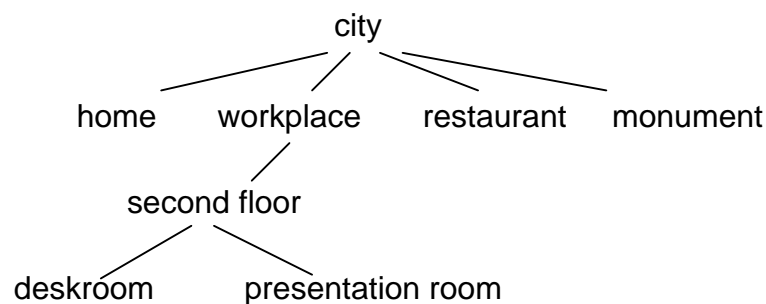


Figure 2. An example of hierarchical arrangements of smart spaces.

1.1.1. Context, Context-awareness, Context-aware Techniques

The definition of an adequate model for context representation represents a crucial step in the process of designing context-aware applications. In the literature the term *context-aware* first appeared in [25], where Dey and Abowd describe context as location, identities of nearby people and objects and changes to those objects. One of the most known definitions is due again to the same authors, who defined context as "any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves" [24]. Several alternate definitions of the term context can be found in literature. A detailed discussion of the differences between them is however out of the scope of this section. For a more comprehensive analysis of the topic we refer the reader to [43].

The generic definition by Dey and Abowd [5] for context and context-awareness is:

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

In the survey, introduced in [6], the Web Service systems were studied by using **context-aware techniques** by answering the following questions:

1. *Context information and context representation*: which techniques are used for modeling context information?
2. *Context sensor techniques*: how context information is measured and sensed?
3. *Context storage techniques*: how context information is stored and how the information is accessed from its storage?
4. *Context distribution techniques*: how context information is distributed and propagated to

IOP principles		9 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

different relevant supporting components? How applications and services can retrieve context information?

5. *Security and privacy techniques*: how context information is protected? Which authentication and authorization mechanism are used for context information? Which techniques are used to ensure the privacy in connection to context sharing?
6. *Context adaptation techniques*: how context information is actually used?

1.1.2. Context-aware architecture

A smart environment is a dynamic system that requires context-awareness implemented as fundamental elements of its architecture. Architecture, i.e. a system, is context-aware if it can extract, interpret and use context information and adapt its functionality to the current context of use [2, originally in 3]. Context-aware systems have been classified based on the typical architecture layers of distributed systems; network layer at the bottom and middleware, application and user interface layers forming a layered stack [2]. The classification framework of context-aware systems consists of five layers: a concept and research layer, network infrastructure layer, middleware layer, application and service layer, and user infrastructure layer (Figure 3) The main interest from the context-aware IOP point of view is in the research layer; context-aware (meta-)modeling, context reasoning, the use of agent and service technologies for context-awareness, context-awareness design and evaluation, security and privacy and first of all, context data management.

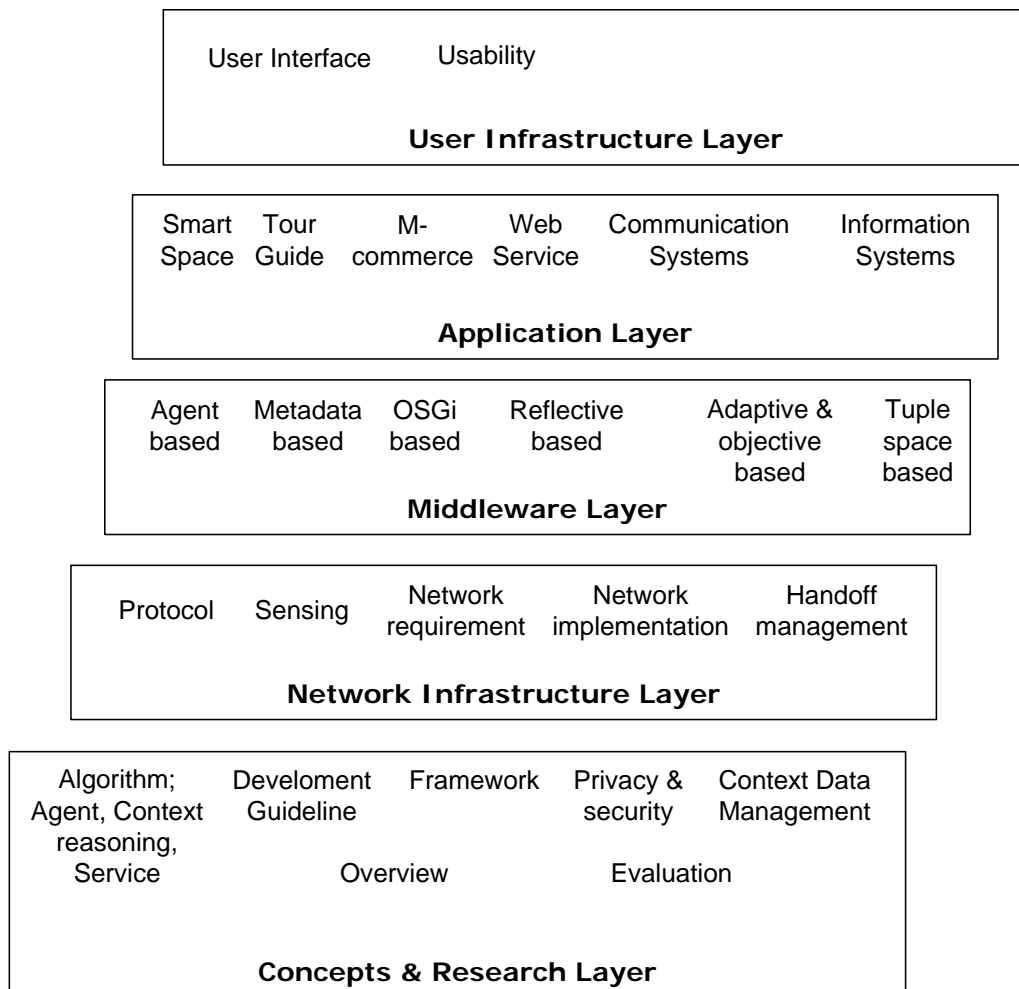


Figure 3. The classification framework of context-aware systems [3].

IOP principles		10 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Throughout this section we will survey the most relevant context modeling approaches, by classifying them based on the data structure scheme used to represent and share contextual information in the systems where such models were defined.

- **Key-value pairs.** The model of key-value pairs is the simplest data structure for modeling contextual information. Schilit et al. [41] used key-value pairs to model the context by providing the value of a context information, e.g., location information, to an application as an environment variable. The key-value modeling approach is frequently used in distributed service frameworks, such as discovery frameworks, e.g., Jini [5] or SLP [8], where service functionalities are described with a list of simple attribute-value pairs, and the discovery procedure operates an exact matching on these attributes. Similarly to service attributes, in those systems context information is described in terms of attribute-value pairs. Key-value pairs are easy to manage, but lack capabilities for sophisticated structuring to enable efficient context retrieval algorithms.
- **Markup scheme.** This approach defines a hierarchical data structure consisting of markup tags with attributes and content. In particular, the content of the markup tags is often recursively defined by other markup tags. Markup-based context representations usually exploit a derived language of the Standard Generic Markup Language (SGML) [7], the superclass of all markup languages, to serialize context information. The most commonly adopted language is the eXtensible Markup Language (XML) or one of its vocabularies [18]. This kind of context modeling approach typically requires the specification of profiles (for a detailed discussion on profiles, see Section 2.2).
- **Graphical model.** A very well known general purpose modeling instrument is the Unified Modeling Language (UML) which has a strong graphical orientation (UML diagrams). Due to its generic structure, UML is also appropriate to model the context. A relevant example is the graphic-oriented context model introduced in [31] by Henriksen et al., which is a context extension of the Object-Role Modeling (ORM) approach [30] according some contextual classification and description properties. A graphical approach to context modeling is particularly suited to database-oriented applications, for example to derive Entity-Relationship (ER) context models and store them into relational databases.
- **Object-oriented model.** Object-oriented approaches to context modeling aim at exploiting the main benefits of this approach, namely encapsulation and reusability, in the process of representing and accessing context in ubiquitous environments. By relying on the abstraction of *object*, they encapsulate and hide from external access the details about context collecting and processing, while providing contextual information by means of interfaces. For example, the concept of *cues* developed within the TEA project [42] provides an abstraction for physical and logical sensors: in particular, a cue represents a function, which takes as an input the value of a single physical or logical sensor at a certain time, and provides as an output the symbolic representation of a certain context.
- **Logic-based models.** In a logic-based context model, context is represented and processed by means of facts, expressions and rules. Generally speaking, a logic defines the conditions on which a concluding expression or fact may be derived (a process known as reasoning or inference) from a set of other expressions or facts. To describe these conditions, a set of rules a formal system is applied. Contextual information is represented by means of logical expressions. It is added, updated and (possibly) deleted from a logic based system in terms of facts, or inferred from appropriate rules defined in the system. All logic based models share a rather high degree of formality in context representation and processing. For example, the Sensed Context Model proposed by Gray and Salber [27] exploits first-order predicate logic as a formal representation of contextual propositions and relations. Another approach within this category is the framework GAIA [38]. Other solutions adopt additional logics, such as for instance fuzzy logic, to represent and reason about uncertain context information or determine the quality of context information [16]. Let us

IOP principles		11 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

note that the exploitation of logics in context representation allows automated reasoning over context information.

- **Ontology-based models.** According to Gruber's definition, an ontology can be defined as "a formalization of a conceptualization" [28]. In a semantic approach, ontologies allow the description of context within specific knowledge domains by means of explicit formalisms, which can be used to represent and reason about context information. Semantic-based context models represent an emerging approach in context representation since they support knowledge sharing and reuse, and logical inference capabilities. Relevant examples include the CONON context modeling approach by Wang et al. [**Error! Reference source not found.**] and the SOUPA ontology developed within the CoBrA system [21]. These approaches will be discussed in more detail in the following section, which provides some insights on Semantic Web technologies and their exploitation for context representation and reasoning.

1.1.3. Metadata-Based Context Representation

An emerging approach to support context awareness and to perform application management accordingly is the adoption of *metadata* for representing both context information and the choices in application behavior at a high-level of abstraction, with a clean separation between application management and application logic. Metadata can describe both the structure/meaning of the resources composing a system and the specification of management operations expressed at a high level of abstraction [37].

The effectiveness of the metadata adoption depends on the characteristics of the language used for metadata specification and of the runtime environment for the metadata support. Metadata specification should exploit declarative languages to accommodate users of different expertise, to simplify metadata reuse and modification, and to facilitate the analysis of potential conflicts and inconsistencies. Metadata runtime support should be responsible for metadata distribution/update and for policy activation/deactivation/enforcement, independently of application logic. The following sections will first provide an overview of emerging metadata specification models, and will then present some relevant examples of existing middleware solutions that exploit metadata to enable the context-aware adaptation of mobile applications.

1.1.3.1. Profile Modeling and Representation

Among the different possible types of metadata, profiles are considered of increasing interest and start to be widely exploited in open and dynamic distributed systems. *Profiles* represent characteristics, capabilities, and requirements of users, devices, and service components. For example, markup scheme and ontology-based context representation models are typically encoded as profiles.

Several research efforts are attempting to identify well accepted formats for the most common access devices and spreading standard profile adoption for expressing user needs/requirements. Profile standardization is in fact crucial for resource reusing and sharing in pervasive environments. Most common examples of profiles include *user* and *device* profiles. The former usually describe data about user preferences, interests and demographics, as well as behavior models. The latter generally contain technical data describing device capabilities, such as available memory, screen resolution and installed software, as well as device status parameters, e.g., battery level. Modeling the context of mobile applications also requires considering profiling parameters of the *environment*, such as properties of the network connection between the user and the accessed service, and conditions of the user's environment, e.g., light, temperature and weather conditions. In addition, since most mobile applications are designed according to a service-oriented approach, it is also necessary to provide support for *service* profiling. [11]. We do not intend to provide here a comprehensive survey about the state-of-the-art profile modeling solutions, but to outline main research directions and existing standards for profile specification.

Several research efforts and commercial solutions model and exploit user profiles. According to [11], they can be classified by taking into account different dimensions, including the modeling approach to user profiles, the richness and generality of user data included in the model, and the method to acquire

IOP principles		12 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

data from the user, e.g., by means of explicit user input or by deriving information from user behavior. We do not focus here on the issue of collecting information from the user, instead we refer to the case when information is explicitly gathered from the user, e.g., by direct user input, and exploited to perform some kind of tailoring of applications based on the defined profile. Several systems adopt XML-compliant formats for user profile specification, while others define ontology-based user profiles [21, 10]. For example, the CARE framework, which originally supported only CC/PP-compliant profile data, has been enhanced to allow the specification of ontology-based profiles, encoded in OWL-DL [10]. This allows choosing the most suitable option for profile representation, depending on whether the application needs more expressive power (mainly provided by OWL) or efficiency (favored by the exploitation of a simpler format like CC/PP). Most systems define their own model of user profiles, and a comprehensive description of existing solutions is out of the scope of this section. Let us note that, although those models tend to be similarly structured, until now no wide agreement on a common standard for user profiling has been reached yet. Some promising solutions are emerging in the field of Semantic Web, which characterize the user in terms of his social relations. For example the Friend-of-a-Friend (FoaF) project describes a user's social network by means of a dedicated RDF ontology [4].

The most prominent solution in device profiling is the Composite Capability/ Preference Profiles (CC/PP) standard, defined by the World Wide Web Consortium (W3C) [35]. CC/PP exploits the XML serialization of RDF to allow the creation of profiles describing the capabilities of a device and possibly the preference of its user. CC/PP profiles are structured as sets of components that contain various attributes with associated values. Components and their values are defined in CC/PP vocabularies, specified in RDF, such as the UAProf vocabulary proposed by the Open Mobile Alliance for representing the hardware, software and network capabilities of mobile devices [9]. Let us note that CC/PP, whose first version was recommended as a W3C standard in 2004, is now in the process of being upgraded to benefit from the functionalities of the newer version of RDF. The main advantage offered by CC/PP lies in the great standardization effort that was undertaken by several mobile device manufacturers to reach an agreement on a common format for the representation of device characteristics. However, CC/PP does neither support user nor environment profiling, which hinders its widespread adoption for context and metadata representation.

As far as service profile modeling is concerned, significant efforts have been spent both by academia and industry to define a common representation format for describing services. In particular, the Web services community has been promoting XML-based standard profiles and protocols to describe, search and retrieve services on the Web. The Web Service Description Language (WSDL), which is now a W3C standard, represents the most significant solution for service profile specification and has been in fact adopted by many companies to allow the development of Enterprise Application Integration solutions [22]. WSDL mainly describes a service in terms of expected input and output, where inputs and outputs are represented by messages, and it also provides a reference, the so-called *grounding*, to the concrete implementation of a service instance. To provide more expressive representation models for services, several semantic-based languages for service description have been proposed, such as OWL-S [26], WSMO [40] and Meteor-S [45], which model both service interface (input/output) and service process workflow by relying on service ontologies.

1.1.4. Context Management and Provisioning in Mobile Environments

Several architectures have been developed to address the issue of collecting, managing and distributing context information to interested applications. Different approaches can be adopted depending on specific application requirements and scenarios, such as the deployment of sensors, the number of potential system users, as well as the technical properties of used devices.

Until now, different categorizations of context management systems have been proposed, none of them being neither exhaustive nor fully agreed on. The proposed classifications focus on different issues related to context management, such as context acquisition, access and sharing, as well as on architectural

IOP principles		13 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

properties of the context management support system. In the following we provide an overview of three relevant state-of-the-art proposals that classify context management systems according to different criteria. It is worth noting that the classification principles characterizing each proposal cannot be considered orthogonal nor clearly distinguished. However, because of the lack of agreement on a common conceptual classification, hereinafter we present them separately.

As far as context acquisition is concerned, Chen presents three possible approaches [20]:

- **Direct sensor access.** This approach is often used in devices with locally built in sensors. The client software gathers the desired information directly from these sensors, which means that there is no additional layer for gaining and processing sensor data. Sensor drivers are hardwired into the application. This tightly coupled approach is not particularly well suited for distributed systems due to its lack of flexibility and reusability.
- **Middleware.** A middleware-based approach introduces a layered architecture in the design of context-aware systems with the intention of hiding low-level sensing details. The middleware is responsible for collecting context information from sensors, storing and aggregating it, and distributing it to interested applications. Compared to direct sensor access, this technique simplifies application development since the client application code does not depend on specific sensors, and it favors the reusability of sensing components that encapsulate sensors.
- **Context server.** The presence of a server permits multiple clients access to possibly remote context data sources. This distributed approach extends the middleware based architecture by introducing a remotely accessible component, the so called *context server*, which gathers sensor data and makes them available to client applications via concurrent, multiple accesses. Beside the reuse of sensors, the usage of a context server has the advantage of relieving clients from the burden of performing resource-intensive operations. As a drawback, the design of a context-aware system based on client-server requires to consider issues like communication protocols, network performance, quality of service parameters, which characterize a client-server distributed system.

Another interesting classification of context management systems is to be found in[48], where Winograd describes three different context management models for coordinating multiple processes and components. This classification is focused on context access and distribution rather than context acquisition.

- **Widgets.** Derived from the homonymous graphical user interface elements, a widget is a software component that provides a public interface for a hardware sensor[25]. Widgets hide low level details of sensing and ease application development due to their reusability. Widgets are usually controlled by a manager. The tight coupling of widgets with their managers increases efficiency, but leads to a lack of tolerance to component failures.
- **Networked services.** This more flexible approach resembles the context server architecture described above. Instead of a global widget manager, discovery techniques are used to find networked services. This service based approach is not as efficient as a widget architecture due to complex network based components, but provides increased robustness and scalability.
- **Blackboard model.** In contrast to the process-centric view of the widget and the service-oriented model, the blackboard model represents a data-centric view. In this asymmetric approach, applications post messages to a shared media, the so called *blackboard*, and subscribe to it to be notified when some specified event occurs. Advantages of this model are the simplicity of adding new context sources and the easy configuration. Unfavorable is the need of a centralized server to host the blackboard and the lack in communication efficiency as two hops per communication are needed.

Finally, Hong and Landay propose a classification for software systems to support context-aware applications [33]. In this case, by focusing on the architectural properties of the context management support, they outline four main categories.

IOP principles		14 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

- **Libraries.** A library is a generalized set of related algorithms. Libraries are mainly developed to promote code reuse. For example, implementations of the JSR 179 Location APIs for Java 2 Micro Edition include code for manipulating location information within the Java framework [6]. Libraries are generally lightweight and easy to use. However, they tend to be focused on low level context details and do not provide any support for application design.
- **Frameworks.** With respect to libraries, a framework-based approach is more focused on design reuse by providing a basic structure for a certain class of applications, which can be customized according to the application requirements. A relevant example falling into this category is the Java Context Aware Framework (JCAF), a Java-based lightweight infrastructure and programming API, developed by Bardram et al., to support context-aware applications [13]. The aim of JCAF is to let programmers focus on modeling and using context information specific for their application, while relying on a basic infrastructure to handle the actual management and distribution of this information. The main limitation of JCAF lies in the fact that it is bound to a specific programming language and environment, i.e., J2ME, which is not supported by most portable devices, and depends on pre-defined communication protocols that cannot be altered, thus leading to a lack of flexibility in system implementation. In addition, the framework defines its own high level model of context: on one side this might help in the design of a context-aware application, on the other side it prevents the design of possibly needed extensions to the context model itself.
- **Toolkits.** Toolkits are typically built on frameworks and offer a number of reusable components each one addressing a specific functionality. For example, a toolkit might offer reusable components for accessing sensors and aggregating context information, such as in the case of the well known Context Toolkit from Dey and Abowd, the first comprehensive support toolkit for context-aware applications development [25]. Toolkits represent a significant step towards the realization of reusable context management support systems. However, similarly to frameworks, they typically depend on specific implementation platforms, operating systems and/or programming language. The Context Toolkit already supports some kind of interoperability. The main limitation of a toolkit is therefore its lack of network-based access: implemented as a single application, a toolkit does not provide any support for distributed access and sharing of context information.
- **Infrastructures.** An infrastructure represents a well-established, reliable and accessible set of technologies acting as a foundational basis for other systems. In particular, service infrastructures expose their capabilities as *services*, which can be generally defined as logical units of functionality, usually accessible via a network. Let us note that middleware support solutions are typically implemented as infrastructures. Several infrastructures have been proposed to support context-aware applications. Middleware infrastructures are particularly well suited to support the development of context-aware applications, as we will explain in the following section.

As stated before, the different categories defined in the above classifications might partially overlap. For example, the concept of infrastructure, as described in [33], clearly resembles the networked services defined by Winograd in [48]. In addition, some context acquisition and distribution models are naturally suited to be exploited within some architectural approaches, such as in the case of widgets, which represent the basic components of the Context Toolkit by Dey and Abowd [25].

In general, the choice of an adequate model for context acquisition and management depends on specific application requirements and characteristics, however, we believe that an infrastructure-based approach relying on the underlying network support for distributing and accessing context bring several advantages. First, by providing uniform abstractions and reliable services for common operations, middleware infrastructures facilitate the development of robust applications even on a diverse and changing set of devices and sensors. In addition, they allow sharing and reuse of context information, while carrying the burden of data acquisition, processing and interoperability on behalf of the application. Finally, the modular nature of a middleware infrastructure allows customizing context provisioning and management depending on the specific needs of the client application.

IOP principles		15 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

In the following section, we will describe some relevant existing solutions in the area of middleware infrastructures for context provisioning. A more extensive description can be found in [12].

1.1.5. Context Management Middleware Infrastructures

The most common design approach for distributed context management frameworks is a hierarchical infrastructure with several components organized in a layered architecture, as depicted in Figure 4. In particular, the *Raw Data Retrieval* layer collects data from sensors on the underlying layer. Let us note that a sensor can be thought of as a programmatic interface that, when queried, returns an answer about a specific context data. This means that, beyond physical sensors, e.g., light and temperature sensors, software applications might also serve as sensors, such as in the case of a user's personal calendar providing information about the user's current activity. The *Preprocessing* layer includes any processing component that takes raw context as an input, performs a data processing activity and provides processed data as an output. Examples of processing activities include context aggregation and verification, as well as inferring additional context information from available data. Once processed, context information is stored and managed by the dedicated layer for further retrieval and access by context-aware applications, which are logically layered on top of the architecture and exploit the underlying components to be provided with needed context information. Let us note that the same logical architecture can be implemented and deployed according to different schemas, e.g., as a centralized server or a distributed system, thus achieving variable levels of efficiency and scalability.

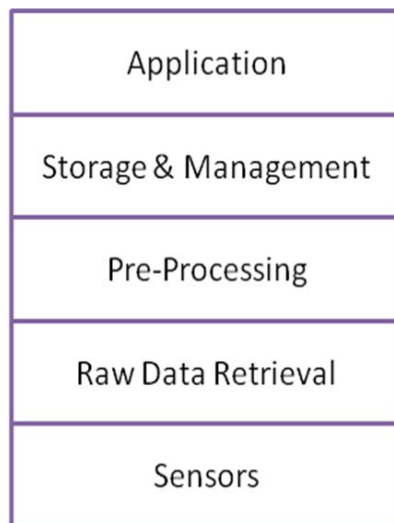


Figure 4. Context Management Layered Infrastructure.

Despite being, strictly speaking, a toolkit, the Context Toolkit represents the first relevant example of infrastructure-oriented support solution for context management [23]. Therefore, its proposed model exerted a prolonged influence on subsequent research in the area of context-aware computing. The system is based on a centralized discovery server where distributed sensor units (called *widgets*), interpreters and aggregators are registered in order to be found by client applications. The toolkit also provides object-oriented APIs to create instances of these components. These components and their respective functionalities set a reference for further research and the layered architecture of Figure 4 was developed based on this original model. The SOCAM (Service-Oriented Context-Aware Middleware) project introduced by Gu et al. proposes an architecture for the building and the rapid prototyping of context-aware mobile services [29]. It relies on a central server, called context interpreter, which acquires context

IOP principles		16 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

data through distributed context providers and provides this information after some kind of processing to interested clients.

Another framework based on a layered architecture is presented in the Hydrogen project, whose context acquisition approach is targeted to mobile devices [32]. The architecture consists of three layers: the adaptor layer, the management layer and the application layer, with analogous functionalities to the ones described in the generic layered architecture (see Figure 4). However, differently from most approaches, the Hydrogen system tries to avoid the need to rely on a single centralized server for context acquisition by distributing several context servers on different devices. Devices in physical proximity are in fact enabled to share their contexts in a peer-to-peer manner by exploiting available wireless connectivity options, e.g., 802.11 or Bluetooth. Hydrogen object-oriented context model allows the addition of new context types by specializing the generic context superclass. A notable characteristic of the system lies in the adoption of XML-based formats and protocols for inter-layer communication, thus achieving a certain degree of platform and language independency.

An interesting middleware support solution for distributed context management is Contory [39]. Contory is a middleware specifically designed to accomplish efficient context provisioning on mobile devices. To make context provisioning flexible and adaptive based on dynamic operating conditions, Contory integrates multiple context provisioning strategies, namely internal sensors-based, external infrastructure-based, and distributed provisioning in ad hoc networks. This approach presents two advantages. First, arranging different context strategies permits to compensate for the temporary unavailability of one mechanism and coping with dynamic resource availability. In addition, combining results collected through different context mechanisms allows the application to partly relieve the uncertainty of a single context source and to more accurately infer higher-level context information. Applications can request context information provided by Contory using a declarative query language, which features on-demand, periodic, and event-based context queries.

MobiComp [49] is a context management infrastructure conceived by Prof. Nick Ryan at the University of Kent. Its structure was thought for Cultural Heritage applications, but the principles and the architecture are applicable to every kind of context aware platform. The core element of Mobicomp is the ContextService, a store for context information that enables the coordination of all the components involved in the computational process. The context service stores Context elements which are subject-predicate-object triple, relating an entity identifier to a named context value. Three components exist for interacting with MobiComp: trackers, listeners and aggregators. A tracker is a MobiComp component that acts as a context producer it registers its availability and capabilities by sending appropriate information to the ContextService. Trackers collect raw context data from sensors, such as GPS receivers, and other dynamic or static sources, including configuration files for device capabilities and user-preferences. The initial data are transformed into context elements which are then put into the tuplespace. Listeners can access to MobiComp for the most recent value of a context property or can be subscribed to a certain subset of context elements in order to receive notification of ContextEvents from the ContextService. An aggregator is a MobiComp component that combines the behaviour of both a tracker and a listener. The Aggregator monitors events from the ContextService, rather than a sensor device, and apply a transformation before returning a new element to the tuplespace. Aggregators can combine several low-level sensor elements to produce an element at a higher level of abstraction. For example, temperature, acoustic, light and air sensors can be used to determine environmental quality.

Contextor [61] is a software infrastructure which concretize the ontology presented in [62] the infrastructure presents a set of abstraction levels for context data and a computational model. The lowest abstraction level is the sensing layer in which numeric values are assigned to observables. The transformation layer associate symbols to numeric observables; the situation layer, by using inference and reasoning, identify situations and context from the observables; the exploitation layer adapt information from the infrastructure to the applications and vice versa, context services interact with this layer of the infrastructure.

An other example of context management platform is the Context Awareness Platform (CAP) developed by Telecom Italia TILAB [50]. CAP consists of a Context Broker and a certain number of Context

IOP principles		17 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Providers which provide context based services to the users. The platform can be accessed by software modules called Context Consumers and Context Sources. Context Consumers ask the CAP for context data, while Context Sources provide context to the platform. The platform also includes a cache for fast retrieval of recent context data and a Context History data base, used for data mining and clustering activities. The language to represent and transport data is called *ContextML* and it provides uniform representation of data as well as interoperability between platform modules.

Some of the most important providers implemented in the platform are the location provider and the situation provider. The location provider, starting from raw data provided by GPS receivers or from operators cell-IDs, can abstract the information about location to a human readable format (reverse-geocoding). The situation provider is a software module which computes different context parameters through a set of rules managed by a rule engine. When a situation for which the user subscribed a service is detected, the service can be provided proactively. The CAP has also an exposure layer that interfaces towards third party services (e.g. the calendar provider has access to Google Calendar).

1.1.6. Ontology-based Approaches to Context Representation and Reasoning

Semantic languages have gained considerable attention within the pervasive research community as a suitable means to provide expressive context representation, querying and reasoning support [21,46,36]. Compared to alternate representation models, semantic-based approaches are emerging because of the several advantages they bring in context modeling. Semantic languages permit to describe at a high level of abstraction the structure and properties of the entities composing a pervasive system, e.g., users, devices and resources, and the desired management operations to govern and control entity behavior. These features appear to be particularly attractive in ubiquitous environments characterized by constantly changing context conditions. The adoption of ontologies to describe context in pervasive computing scenarios brings several advantages by allowing the exchange of semantics about the described context, it enables mutual understanding between previously unknown entities about their capabilities and the current execution context. Moreover, Semantic Web languages enable expressive querying and automated reasoning over context representation, to derive additional and/or higher level context information that can be exploited by the application.

In this section, we first provide an overview of Semantic Web standard languages, followed by a description of relevant existing work on ontology-based context representation models. Particular attention is given to SOFIA-related application domains, i.e., personal space, smart home and smart city scenarios.

Over the last decade W3C has been working on standard specifications for Semantic Web languages. The Resource Description Framework (RDF) can be considered the very first representation model provided for the Semantic Web (and to a certain extent created together with the original concept of the Semantic Web). Further efforts have been spent to accommodate requirements for greater expressivity and reasoning power, which led to the official recommendation of the Web Ontology Language (OWL). It is worth noting that several other language/representation model proposals are emerging beside RDF and OWL, as well as proposals for adding properties and features to existing standard languages.

1.1.6.1. Resource Description Framework

Semantic Web technologies can be thought as a layered framework, whose lower layers provide data interchange formats, both syntactic and semantic, on top of which ontologies can be build, queried and possibly supplemented by rules, as shown in Figure 5 (<http://www.w3.org/2007/03/sw>). In particular, the Resource Description Framework (RDF) is a language originally created for representing information about resources in the World Wide Web [34]. By generalizing the concept of a Web resource, RDF can also be used to represent information about resources that cannot be directly retrieved on the Web, including user preferences, mobile device properties and any other context information. It is worth noting that RDF provides a common framework for expressing the semantics of this information so it can

IOP principles		18 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

be exchanged between applications without loss of meaning. RDF identifies things using Web identifiers (called Uniform Resource Identifiers, or URIs), and describes resources in terms of simple properties and property values. In particular, RDF represents simple statements about resources as a graph of nodes and arcs representing the resources, and their properties and values. For example, the statement "Dave Beckett is the editor of the resource <http://www.w3.org/TR/rdf-syntax-grammar>" can be represented by the graph depicted in Figure 6 (<http://www.w3.org/TR/rdf-syntax-grammar/#section-Syntax-node-property-elements>). To encode RDF statements in a machine-processable way, RDF relies on a serialization based on the Extensible Markup Language (XML) [18].

RDF properties may be thought as attributes of resources, but may also represent relationships between resources. RDF however, provides neither mechanisms for describing these properties, nor does provide mechanisms for describing the relationships between these properties and other resources. To overcome this limitation, some extensions such as the RDF Schema (RDF-S), i.e., the RDF vocabulary description language, have been defined [19]. RDF-S defines classes and properties that may be used to describe classes, properties and other resources, such as the domains and ranges of properties.

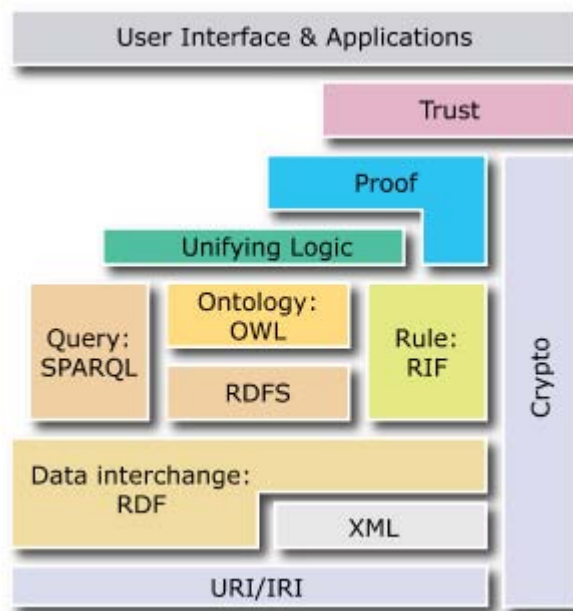


Figure 5. Semantic Web Layered Architecture (www.w3.org).

1.1.6.2. Web Ontology Language

The first level above RDF required for the Semantic Web is an ontology language that can formally describe the meaning of terminology used in "Web resources", as described above. The Web Ontology Language (OWL) provides an expressive vocabulary for describing properties and classes: among others, relationships between classes, such as disjointness, cardinality, equality, richer typing of properties, characteristics of properties, such as symmetry, and enumerated classes [14].

OWL is divided to sublanguages: OWL Lite, OWL DL and OWL Full – and these sublanguages are upward compatible. Thus, OWL Lite is legal OWL Full ontology[16] Following paragraphs present OWL's sublanguages briefly.

OWL Lite supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives.

IOP principles		19 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

OWL DL supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class).

OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

Let us note that other languages have been proposed and used to define ontologies beyond Semantic Web languages. For example, it is possible to define ontologies by means of well-established logic languages, such as Logic Programming languages as Prolog. However, Semantic Web languages offer the great advantage of providing interoperability and standardization, based on the XML format of serialized RDF or OWL documents. In addition, their increasing success within the research community, both from academia and industry, represents a promising step towards the reach of a shared agreement on semantic interoperable standards.

In this section we present some relevant approaches to context modeling that are based on ontologies. Since many context ontology solutions have been proposed, we hereinafter focus on early stage and most significant efforts.

A significant context modeling approach based on ontologies is the CoBrA system [20]. The CoBrA system uses a broker-centric agent architecture to provide runtime support for context-aware systems in ubiquitous computing environments, such as intelligent meeting rooms. CoBrA relies on the SOUPA ontology, which provides a set of ontological concepts to characterize entities such as persons, places or several other kinds of objects within their contexts. SOUPA is developed in OWL.

Another interesting approach has been proposed as the Aspect-Scale-Context Information model [44]. Differently from SOUPA, this model provides its own Context Ontology Language (CoOL), which is supplemented by integration elements such as scheme extensions for Web Services. The CoOL language is used to support context-awareness in distributed service frameworks for various applications, like for example to check service interoperability in terms of contextual compatibility.

The CONON context modeling approach aims at developing a context model based on ontologies to exploit knowledge sharing, logic inferencing and knowledge reuse capabilities [47]. Similarly to CoBrA, Wang et al. created an upper ontology which captures general features of basic contextual entities and a collection of domain specific ontologies and their features in each sub-domain. The CONON ontologies are serialized in OWL-DL. This allows for consistency checking and contextual reasoning using inference engines developed for description logic-based languages.

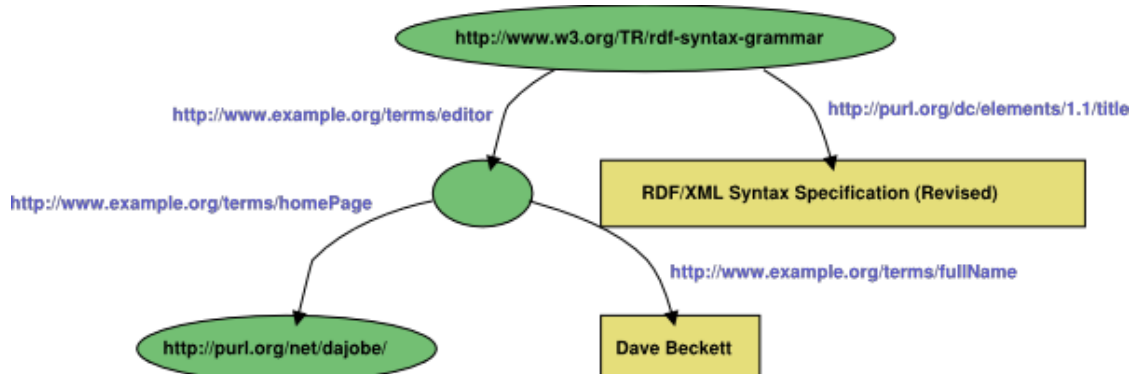


Figure 6. RDF Graph Example (www.w3.org).

IOP principles		20 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

1.1.7. Existing Context Ontologies for SOFIA Application Domains

In recent years several context ontologies have been developed as part of different research efforts. On one side, such rapid development demonstrates the growing interest towards ontology-based technologies to represent and reason about context; on the other side, the lack of a common effort to standardize context ontologies (at least for some well known application domains, such as the ones addressed in SOFIA) led to a plethora of similar and partially overlapping ontologies, where one specific solution fails to clearly emerge.

For example, smart housing is a very active research field, as demonstrated by several existing research efforts, e.g., Georgia Tech Aware Home, Gator Tech Smart House, and the FP6 Ambient Intelligence for Networked Home Environment (Amigo) Project (<http://www.hitech-projects.com/euprojects/amigo/>). While those efforts mainly focus on the design of architectures to collect context from ubiquitous sensors and to consequently take house management/configuration decisions, to the best of our knowledge, there is no proposal that provides with complete, publicly available, and reusable Smart Housing Ontologies (SHOs). The few SHO-related researches provide only partial ontologies, focused on modeling (i) information about available services and SOA components, their interface, and mutual interactions or (ii) context specific to the targeted application domain, such as data about light/temperature. For instance, STREP FP6 SWOP has defined a home maintenance ontology, which could be used as a starting point to describe some home appliances. Another recent smart home-related ontology is the Dog-Ont ontology, which however, as many others, has been designed with the specific purpose of supporting smart home environment, thus lacking a general and systematic approach [17]. In addition, most SHO ontologies are not publicly available.

More general ontologies for smart spaces, such as ULCO and SOUPA (mentioned above), have been developed for partially related application fields, such as smart meeting rooms and workplace settings. Similar considerations apply for the other application domains addressed by SOFIA in WP1 and WP3, respectively.

The Mobile Ontology, developed in the European IST-FP6 project SPICE[63], is a higher-level comprehensive ontology for the mobile communications domain, it is divided into sub-ontologies for the different domains covered e.g. context, profile, presence and services.

Dolce: (Descriptive Ontology for Linguistic and Cognitive Engineering)[64] aims at capturing the ontological entities underlying natural language and human commonsense; its cognitive bias could be very important when considering the global trend towards the Semantic Web concretization.

Very recent research initiatives, such as the W3C Workshop on the Future of Social Networking (held in January 2009), have started to recognize the need to build common context ontologies, in order to easily integrate contextual information with socially oriented applications, especially running on mobile phones and other handheld devices. This calls for the development of a context ontology relative to the user rather than absolute, e.g., including terms to annotate a location as "home" and "office" rather than as a postal address or a geo-location, or a time as "workday" or "commuting time" rather than "Monday" or "8am" (<http://www.w3.org/2008/09/msnws/report#Context>). It is worth noting that similar research efforts also account for the emerging trend towards integration of context aware and socially aware mobile applications.

Whatever existing approach/solution might be chosen, context ontologies should be developed according to well known patterns and methodologies. In particular:

1. general ontology development methodologies, such as OntoClean, and related development tools (<http://www.ontoclean.org/>)
2. ontology development methodologies specific for the considered application domain, i.e., personal space (WP1), smart home (WP2) and smart city (WP3). For example, the eCognos methodology, which includes foundational ontologies (<http://www.e-cognos.org>), could be adopted for smart housing ontology development.

1.1.7.1.

Sensor Ontologies

Sensor ontologies are an important subset of context ontologies and some research has been done to detect the most important standards to which relate the entities.

One of the most cited ontology for the description of sensors and their data is OntoSensor[51].

As good ontological engineering includes attempts to leverage upper ontologies where domain independent knowledge is captured, OntoSensor extends the IEEE SUMO[52] upper-level ontology: OntoSensor classes extends some of the classes defined in SUMO. OntoSensor is also based on the International Organization for Standardization (ISO) 19115 which defines the schema required for geographic information and services. OntoSensor can be viewed as a middle-level ontology because it extends the concepts of a high-level ontology (SUMO) and its concepts can be used by more specialized ontologies that model domain-specific sensors. An other example of sensor ontology is proposed in [53]: again SUMO is taken as the starting point, but the organization of the classes is different. In particular three ontologies that extend SUMO have been proposed: the Sensor Hierarchy Ontology (SHO) includes knowledge models for transducer (sensors and actuators) elements; the Sensor Data Ontology (SDO) describes the dynamic and observational properties of transducers; the Extension Plug-ins Ontologies (EPO) extends the capabilities and behaviour of the others by allowing developers to integrate domain-specific ontologies.

1.1.8. **Event-based communications**

Event-based communications based on the publish/subscribe paradigm [54] have been introduced in middleware platforms to provide an intuitive reactive asynchronous communication paradigm that can be exploited to realize sense-and-react applications (e.g. air traffic control system, defence systems, fraud detection systems etc). This kind of applications needs to response in some milliseconds to changing conditions in the environment. This can be achieved if the sensing portion of the application is able to poll the environment rather than having data pushed to it [55]. Timeliness and reliability of data delivery are essential to guarantee the correctness and safety of an application (and application users) in such a domain. If the system fails in delivering data on time, threats may be realized possibly resulting in a dangerous situation for either infrastructure or human lives.

An event-based interaction assumes that the interacting parties play the role of either event producers (often called publishers) or event consumers (i.e. subscribers). Consumers can define their interest in a subset of all the produced events by issuing subscriptions that act as filters on the stream of incoming events. A software component, called the event notification service, decouples the interaction between producers and consumers by receiving subscriptions and events, issued by subscribers and publishers respectively, and notifying subscribers about events that satisfy the issued subscriptions. The complete decoupling between interacting parties realized with the mediation of the ENS (??) is a fundamental aspect of this paradigm. Middlewares based on the publish/subscribe paradigm are expected to provide improved interoperability in environments characterized by heterogeneous devices providing asynchronous interaction primitives and standard data models.

An important example of event-based middleware is Java Message Service (JMS) [58]. JMS is a standard promoted by Sun Microsystems to define a Java API for the implementation of message-oriented middleware. The compliance to the specification allows JMS to guarantee portability for Java applications in exchanging messages through products of different vendors. A JMS implementation represents a general-purpose *message oriented middleware* that acts as an intermediary between heterogeneous applications: the applications can choose the communication mode that better suits their specific needs (JMS supports both pub/sub and point-to-point modes). Most of the standards for data distribution middleware, e.g. CORBA Event [59] and Notification [60] services, JMS etc., as well as most proprietary solutions, lacked the support necessary for real-time, mission and safety critical systems. The main drawbacks of these solutions are related to a limited (or not existing) support for QoS and to the lack of architectural properties needed to promote dependability and survivability. Recently, in order to fill this gap, the OMG proposed the Data Distribution Service (DDS) [56] specification. The OMG's DSS for

IOP principles		22 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Real-time Systems is an API specification and interoperability wire-protocol that defines a data-centric publish-subscribe interaction paradigm. DDS is based on a fully decentralized architecture, which provides an extremely rich set of configurable QoS policies to be associated with topics. A publisher can declare the intent of generating data with an associated QoS and writing the data in a topic. Then, it is responsible for disseminating data (in either a reliable or best-effort fashion) in agreement with the declared QoS that has to be compatible with the QoS defined by the topic. The DDS also provides a set of QoS policies in order to control the timeliness properties of distributed data. Specifically, it defines the maximum inter-arrival time for data and the maximum amount of time that should elapse for distribution of data from publishers to subscribers. Note that the above mentioned DDS properties can be effectively enforced only when the DDS is deployed in a strictly controlled setting (i.e., in a managed environment); in a large scale, unreliable and unmanaged contexts, the performance obtainable by the DDS may become unpredictable [57].

Currently, none of the previously mentioned platforms fully supports the smart environment characterizing the SOFIA project. On one hand, some DDS vendors are developing implementations targeted at the deployment on embedded devices, but these solutions are far from being mature products. On the other hand, there are some lightweight implementations of Java Message Service. For instance JORAM (Java Open Reliable Asynchronous Messaging) is available for Java embedded devices. J2ME applications, running on these devices, can use JORAM to cooperate with other JMS applications via Internet. In addition, a JORAM application can employ HTTP/SOAP. The use of the SOAP protocol between a JORAM client and a JORAM server allows non-Java clients to benefit from the JORAM messaging services.

Previously cited platforms can be employed in wireless sensor networks and in safety critical circumstances (like the scenarios presented in UC1 (*Smart Surveillance of Public Areas*) of the D3.11), where every public forces have to be able to interact with the colleagues and the intervention squads through mobile ad hoc networks that are dynamically created among the mobile user terminals. In particular, employing the publish/subscribe paradigm, the actors can subscribe to critical information they want to read and receive only required data. In the same way, people trapped inside a place can be pushed with information about the nearest emergency exit of safe locations through their personal mobile equipments.

Finally, mobile devices running a publish/subscribe application can become publishers of information by sending data about the internal area to the public forces. In this situation, mobile devices act like sensors and share data with other devices of the smart space (UC3 in D3.11).

Event-based middleware and Service-Oriented Architecture have two types of interactions between each other. First, the occurrence of an event can trigger the invocation of one or several remote services. Those services may perform simple functions, or entire business processes. This interaction between events and services is commonly referred to as Event-Driven SOA (ED-SOA). Second, a service may generate an event. The event may signify a problem or impending problem, an opportunity, a threshold, or a deviation. Upon generation, the event is immediately disseminated to all interested parties, using the publish/subscribe system. The interested parties evaluate the event and optionally take action. The event-driven action may include the invocation of a service, the triggering of a business process, and/or further information publication. In this interaction, the service is purely one of many event sources in a broader event-based system. In this way, publish/subscribe systems can be widely employed at the service level of the SOFIA architecture because they can act like triggers for different services of the whole distributed system and, for instance, some DDS vendors provide API to expose their DDS implementation like a Web Services.

1.1.9. References

- [1] I. Marsa-Maestre, M. A. Lopez-Carmona, J. R. Velasco. A hierarchical, agent-based service oriented architecture for smart environments. *SOCA*, (2):167-185, 2008
- [2] A. Dey, D. Salber, and G. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2,3 and 4):97-166, 2001.
- [3] H. Truong, S. Dustdar. A Survey on Context-aware Web Service Systems. <http://www.infosys.tuwien.ac.at/Staff/truong/publications/surveycontextws-submittedversion.pdf>
- [4] Friend of A Friend Project. <http://xmlns.com/foaf/0.1/>.
- [5] Jini Technology. <http://www.jini.org/>.
- [6] JSR 179: Location API for J2ME. <http://jcp.org/en/jsr/detail?id=179>.
- [7] Overview of SGML Resources. <http://www.w3.org/MarkUp/SGML/>.
- [8] Service Location Protocol. Internet Engineering Task Force Request For Comments 2608.
- [9] UAProf Profile Repository, 2008. http://w3development.de/rdf/uaprof_repository/.
- [10] A. Agostini, C. Bettini, and D. Riboni. Loosely coupling ontological reasoning with an efficient middleware for context-awareness. In *MobiQuitous*, pages 175–182. IEEE Computer Society, 2005.
- [11] A. Agostini, C. Bettini, and D. Riboni. *Integrated Profiling of Users, Terminals and Provisioning Environments*, chapter 34, pages 901–937. Auerbach, 2006.
- [12] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.
- [13] J. E. Bardram. The Java Context Awareness Framework (JCAF) - a service infrastructure and programming framework for context-aware applications. In H.-W. Gellersen, R. Want, and A. Schmidt, editors, *Pervasive*, volume 3468 of *Lecture Notes in Computer Science*, pages 98–115. Springer, 2005.
- [14] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference, 2004. <http://www.w3.org/TR/owl-ref/>.
- [15] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli. Context-aware middleware for resource management in the wireless internet. *IEEE Trans. Software Eng.*, 29(12):1086–1099, 2003.
- [16] M. Berchtold, C. Decker, T. Riedel, T. Zimmer, and M. Beigl. Using a context quality measure for improving smart appliances. In *ICDCS Workshops*, page 52. IEEE Computer Society, 2007.
- [17] D. Bonino and F. Corno. Dogont - ontology modeling for intelligent domotic environments. In A. P. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. W. Finin, and K. Thirunarayan, editors, *International Semantic Web Conference*, volume 5318 of *Lecture Notes in Computer Science*, pages 790–803. Springer, 2008.
- [18] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible Markup Language (XML) 1.0 (Fourth Edition), 2006. <http://www.w3.org/TR/2006/REC-xml-20060816/>.
- [19] D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema, 2004. <http://www.w3.org/TR/rdf-schema/>.
- [20] H. Chen. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. PhD thesis, University of Maryland, Baltimore County, Baltimore MD, USA, 2004.
- [21] H. Chen, F. Perich, T. W. Finin, and A. Joshi. Soupa: Standard ontology for ubiquitous and pervasive applications. In *MobiQuitous*, pages 258–267. IEEE Computer Society, 2004.
- [22] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1.
- [23] A. K. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, Georgia Institute of Technology, Georgia, USA, 2000.
- [24] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.
- [25] A. K. Dey, G. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human Computer Interaction (HCI) Journal*, 16:97–166, 2001.

- [26] D. M. et al. OWL-S: Semantic Markup for Web Services. W3C Member Submission, November 2004. <http://www.w3.org/Submission/OWL-S/>.
- [27] P. D. Gray and D. Salber. Modelling and using sensed context information in the design of interactive applications. In M. R. Little and L. Nigay, editors, *EHCI*, volume 2254 of *Lecture Notes in Computer Science*, pages 317–336. Springer, 2001.
- [28] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
- [29] T. Gu, H. K. Pung, and D. Zhang. A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, 28(1):1–18, 2005.
- [30] T. Halpin. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufmann, 2001.
- [31] K. Henriksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. In F. Mattern and M. Naghshineh, editors, *Pervasive*, volume 2414 of *Lecture Notes in Computer Science*, pages 167–180. Springer, 2002.
- [32] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, and W. Retschitzegger. Context-awareness on mobile devices - the hydrogen approach. In *HICSS*, page 292, 2003.
- [33] J. I. Hong and J. A. Landay. An infrastructure approach to context-aware computing. *Human-Computer Interaction (HCI) Journal*, 16(2-3), 2001.
- [34] G. Klyne and J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax, 10 February 2004. <http://www.w3.org/TR/rdf-concepts/>.
- [35] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, and L. Tran. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0, 2004. <http://www.w3.org/TR/CCPP-struct-vocab/>.
- [36] R. Masuoka, B. Parsia, and Y. Labrou. Task computing - the semantic web meets pervasive computing. In D. Fensel, K. P. Sycara, and J. Mylopoulos, editors, *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 866–881. Springer, 2003.
- [37] H. W. Nissen, M. A. Jeusfeld, M. Jarke, G. V. Zemanek, and H. Huber. Managing multiple requirements perspectives with metamodels. *IEEE Software*, 13(2):37–48, 1996.
- [38] A. Ranganathan and R. H. Campbell. A middleware for context-aware agents in ubiquitous computing environments. In M. Endler and D. C. Schmidt, editors, *Middleware*, volume 2672 of *Lecture Notes in Computer Science*, pages 143–161. Springer, 2003.
- [39] O. Riva. Contory: A middleware for the provisioning of context information on smart phones. In M. van Steen and M. Henning, editors, *Middleware*, volume 4290 of *Lecture Notes in Computer Science*, pages 219–239. Springer, 2006.
- [40] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web service modeling ontology. *Applied Ontology*, (1):77–106, 2005.
- [41] W. N. Schilit. *A System Architecture for Context-Aware Mobile Computing*. PhD thesis, Columbia University, 1995.
- [42] A. Schmidt and K. van Laerhoven. How to build smart appliances. *IEEE Personal Communications*.
- [43] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004*, 2004.
- [44] T. Strang, C. Linnhoff-Popien, and K. Frank. Cool: A context ontology language to enable contextual interoperability. In J.-B. Stefani, I. M. Demeure, and D. Hagimont, editors, *DAIS*, volume 2893 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 2003.
- [45] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller. Meteor-s wsdi: A scalable infrastructure of registries for semantic publication and discovery of web services. *Journal of Information Technology and Management, Special Issue on Universal Global Integration*, (6).
- [46] X. Wang, J. S. Dong, C. Chin, S. Hettiarachchi, and D. Zhang. Semantic space: An infrastructure for smart spaces. *IEEE Pervasive Computing*, 3(3):32–39, 2004.
- [47] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung. Ontology based context modeling and reasoning using owl. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and*

IOP principles		25 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Communications Workshops (PERCOMW 2004), page 18, Washington, DC, USA, 2004. IEEE Computer Society.

- [48] T. Winograd. Architectures for context. *Human Computer Interaction*, 16(2-4), 2001.
- [49] Giuseppe Raffa, Philipp H. Mohr, Nick Ryan, Daniele Manzaroli, Marina Pettinari, Luca Roffia, Sara Bartolini, Lukas Sklenar, Franca Garzotto, Paolo Paolini, Tullio Salmon Cinotti *CIMAD - A Framework For The Development Of Context-Aware and Multi-Channel Cultural Heritage Services*
- [50] Marco Marengo, Nicoletta Salis, Massimo Valla. Context Awareness: servizi mobili “su misura”
- [51] David J. Russomanno, Cartik R. Kothari and Omoju A. Thomas. *Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models*
- [52] Ian Niles, Adam Pease. *Towards a Standard Upper Ontology*
- [53] Mohamad Eid¹, Ramiro Liscano², Abdulmotaleb El Saddik¹. *A Universal Ontology for Sensor Networks Data*
- [54] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.
- [55] K. M. Chandy. Sense and respond systems. In International Computer Measurement Group Conference, pages 59–66, 2005.
- [56] OMG Data Distribution Portal, 2008. <http://portals.omg.org/dds>.
- [57] R. Baldoni, L. Querzoni, and S. Scipioni. Event-based data dissemination on interadministrative domains: Is it viable? In FTDCS '08: Proceedings of the 2008 12th IEEE International Workshop on Future Trends of Distributed Computing Systems, pages 44–50, Washington, DC, USA, 2008. IEEE Computer Society
- [58] Sun Microsystem. Java Message Service (JMS). <http://java.sun.com/products/jms/>, 2008.
- [59] Object Management Group. CORBA event service specification, version 1.1. OMG Document formal/2000-03-01, 2001.
- [60] Object Management Group. CORBA notification service specification, version 1.0.1. OMG Document formal/2002-08-04, 2002.
- [61] G. Rey, J. Coutaz. The Contextor Infrastructure for Context-Aware Computing: Component-oriented Approaches to Context-aware Computing held in conjunction with ECOOP'04 in Oslo (14 june 2004)
- [62] Crowley, J. L., Coutaz, J., Rey, G., Reignier, P., "Perceptual Components for Context Aware Computing", UBICOMP 2002, Goteborg, Sweden, (September 2002).
- [63] Spice Mobile Ontology website: <http://ontology.ist-spice.org/>
- [64] Dolce Ontology website: <http://www.loa-cnr.it/DOLCE.html>

1.2. State-of-the-art of Ontology-oriented Security Management

This section concentrates an ontology-oriented security management and will act as a basis for task 5.3 in SOFIA project.

This section is divided into three sub-sections – concentrating to ontology issues. Sections one and two describe security and Quality of Service (QoS) ontologies. Lastly, section three introduces ONTOMETRIC method briefly to facilitate ontology selection.

1.2.1. Security Ontologies

1.2.1.1. Taxonomy of Information Security for Service-Centric Systems

Savolainen et al. present a taxonomy of information security for service centric systems in [3]. The presented taxonomy is intended for the use of software architects of service centric systems. Thus, the taxonomy support following aspects: 1) stakeholders' participation in the development of service-centric systems, 2) improve communication of security concerns in requirements elicitation, 3) aid to designing and constructing of security means while architecting and 4) support quality analysis phase by providing a common security terminology[3].

Security taxonomy is divided to five main classes, i.e. SecurityAssets, SecurityAttributes, SecurityThreats, SecuritySolutions and SecurityMetrics, as shown in Figure 7. The class SecurityAssets contains things that have a value – almost anything can be an asset. In other words, asset means a thing that has to be protected. SecurityAttributes class contains classes Confidentiality, Integrity and Availability, which are also called CIA triad. These three classes compose service's security, and thus, security specification of a system should cover all of these aspects. After that, the class SecurityThreats defines Faults, Errors and Failures. A failure is defined as an event that occurs when the delivered service differ from the correct service. Instead, deviation in the external state of the system is called an error and the cause of an error is called fault. In addition, these faults can exist in internal or external of a system. Internal faults are called vulnerabilities, whereas external faults are called attacks. The SecuritySolutions class contains ControlMechanisms, ControlProperties and FailureResistance subclasses. These security solutions are intended for preventing unwanted behaviour and development of a system. The final class is SecurityMetrics divided to StrengthMetrics and WeaknessMetrics. These classes contain metrics to measure system's threats and their countermeasures. SecurityMetrics class and it subclasses contain eight metrics as an instance format[3]. Table 1 lists fields and their purposes of these instances[4].

Field	Description
Description	Textual description of a metric
MinValue / MaxValue	Minimum / Maximum value that a metric can produce
TargetValue	The optimal value of a metric.
Applicable	When a metric can be applied.
Formula	Formula that is used for calculating metric's value
Annotations	Explanations of terms in the Formula field
Target	Which parts of a system can be measured in this metric (System, Service, Component)

Table 1 Metrics' fields and their descriptions

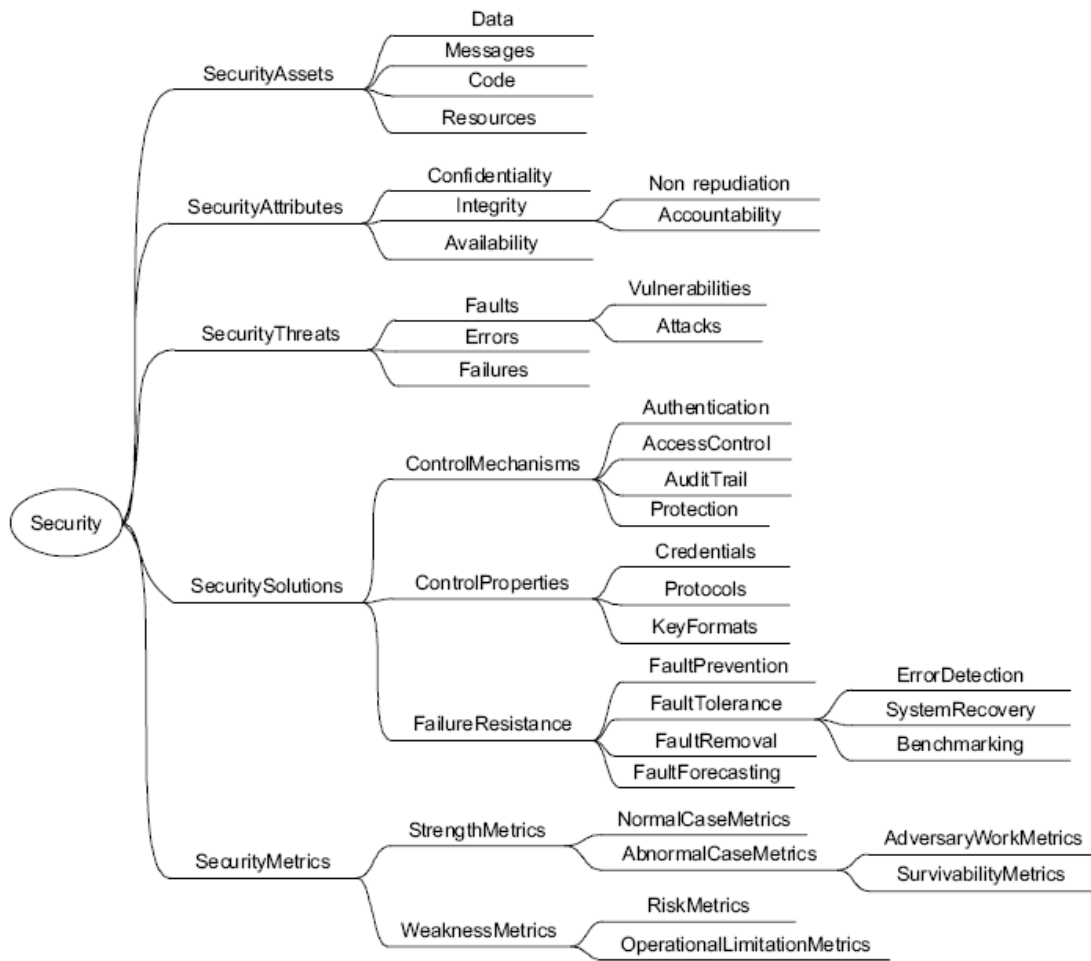


Figure 7 Taxonomy of SOA information security [3]

1.2.1.2. Security Ontology for web services

Denker et al. describe security annotations represented by DAML-S intended to be used by agents in[5]. Their ontology can be used to describe both requirements and capabilities, e.g. requirement that requestor wants to use Open-PGP encryption, and this information are used for service discovery and matchmaking. Based on these requirements and capabilities matchmaking can be performed – four different matching levels are defined: Perfect match, Close match, Possible of negotiation and No match[5]

Figure 8, Figure 9, figure 10 and Figure 11 present ontologies of Denker et al, adopted from [5]. These ontologies contain credentials information and security mechanisms. Therefore, many important security aspects are missing. When compared the work by Savolainen et al., for instance assets, threats and metrics are not defined. However, it looks that purpose of the Denker’s ontology is to concentrate mostly to the service discovery and matchmaking, and thus scope is different than ontology in[3]. Lately, Denker et al. updated their ontology to utilize OWL in [6], but their ontology still concentrates the same use as the previous version.

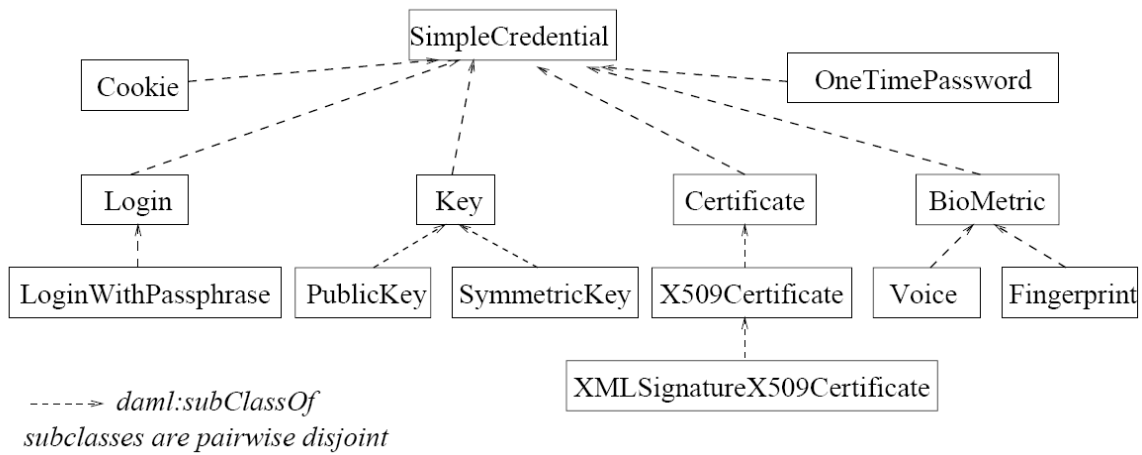


Figure 8 Credentials ontology (1/2)

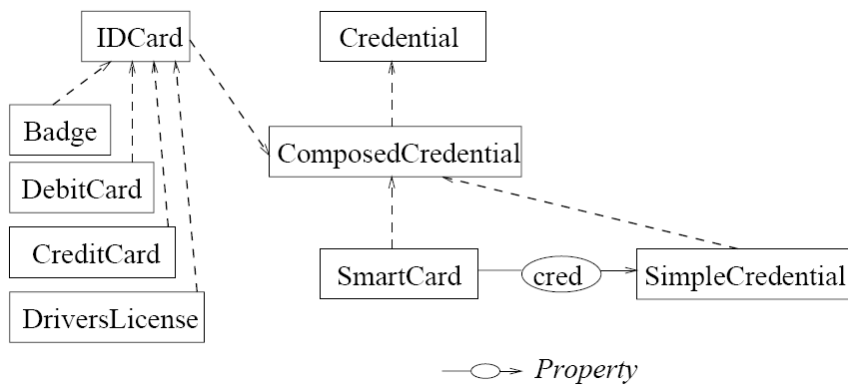


Figure 9 Credentials ontology (2/2)

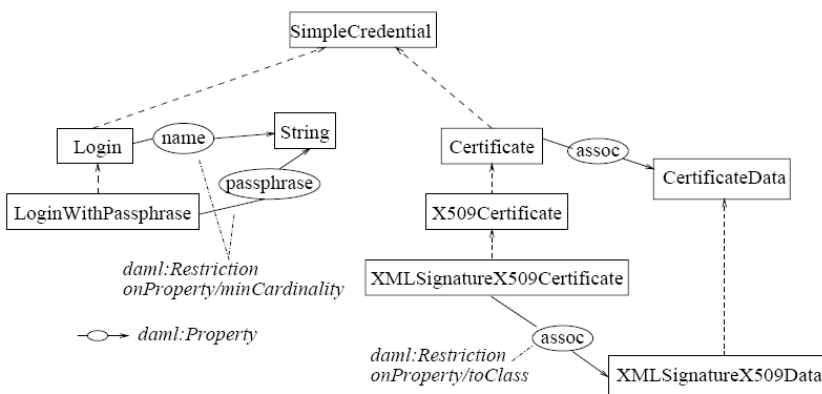


Figure 10 Properties sample from credentials ontology

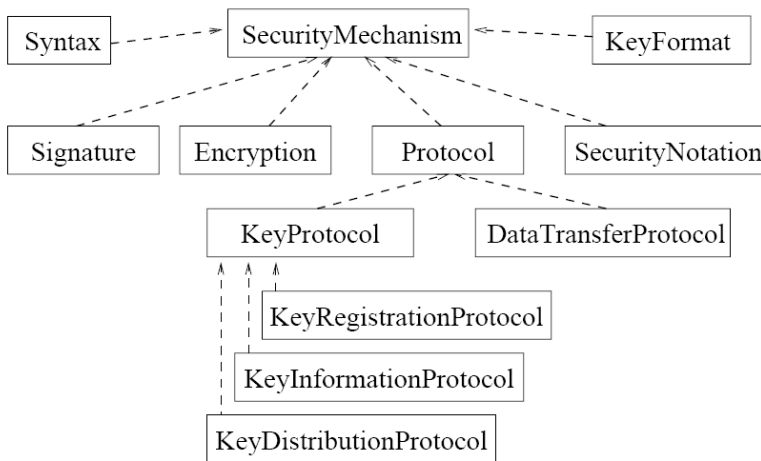


Figure 11 Security ontology

1.2.1.3. Naval Research Laboratory Security Ontology

Kim et al. presented their security ontology called NRL (Naval Research Laboratory) security ontology in [2]. NRL security ontology makes it possible to annotate resources with security related information that can be used during service discovery and matchmaking. Intention of construction of the NRL security ontology was to complement existing ontologies that are focused to annotate functional aspects. NRL security ontology is able to describe following security aspects: mechanisms, protocols, objectives, algorithms and credentials in various levels of details. For instance, requirement for using AES (Advanced Encryption Standard) can be set less or more detailed, by using properties, AES with 256 bit keys. Both service providers and requestors can use NRL ontology to describe their capabilities and requirements.

Actually NRL security ontology is a collection of different ontologies – Figure 12 [2] shows seven separated ontologies that make up the NRL security ontology. Set of grey OWL-S ontologies mean core ontologies used to describe web services. Instead, Service Security, Agent Security and Information Object ontologies are based on existing DAML security ontologies. Thus, the rest four ontologies are made by authors of [2].

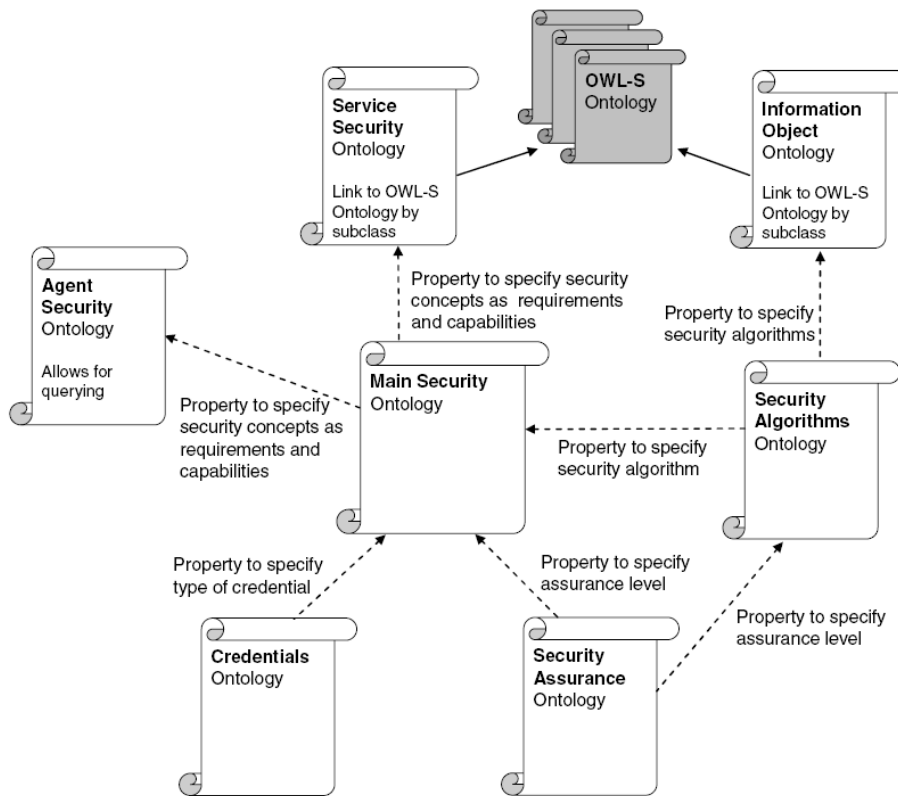


Figure 12 Security-related ontologies, picture adopted from [2]

Main Security ontology – represented in fig. 13 (legend for the figure: ellipses denote classes, solid lines instances of the class and dotted lines means properties) – imports the Credentials, Security Algorithms and Security Assurance ontologies as object properties. Thus, these ontologies make it possible to give more specific values for SecurityConcepts. In addition, user can define SecurityObjectives and connect them to the specific SecurityConcept by utilising supportSecurityObjective property [1]. Whereas, the Service Security ontology makes it possible to use security concepts from the Main Security ontology in the Web services framework. Security-related queries with security concepts – defined in the Main Security ontology – can be made by means of the Agent Service ontology. Lastly, the Information Object ontology makes it possible to annotate inputs and outputs of web service by using the Security Algorithms ontology[2].

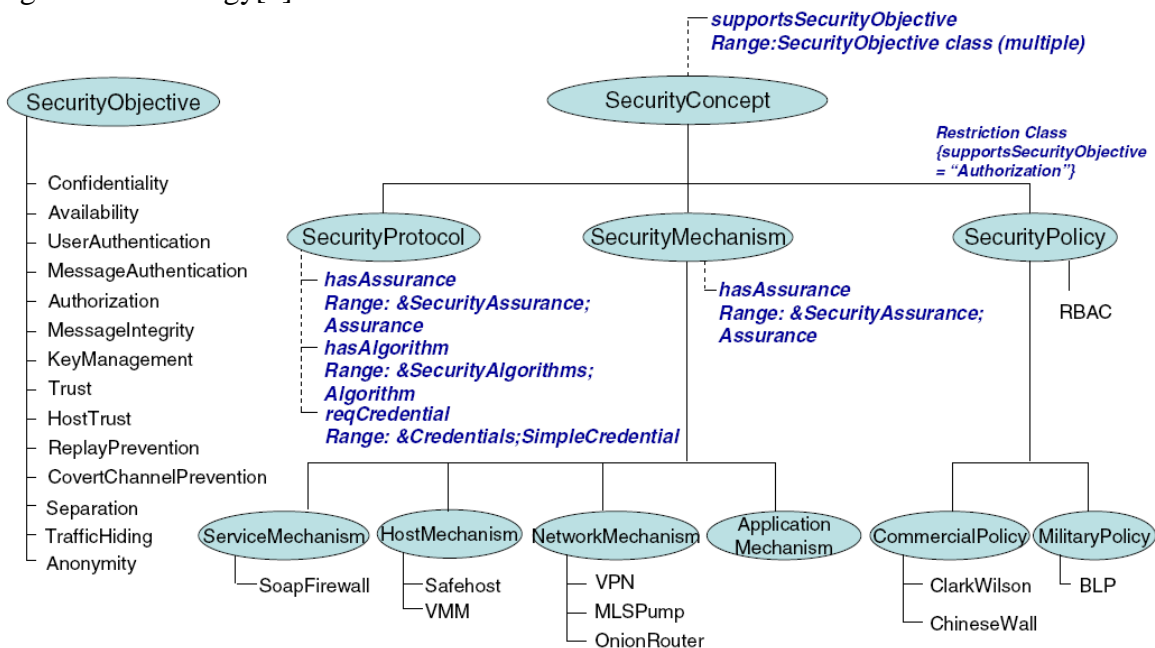


Figure 13 Main security ontology [2]

Following pictures are adopted from the [2] and [1]. Brief descriptions including pictures and related OWL files can be found also from the NRL web page [1].

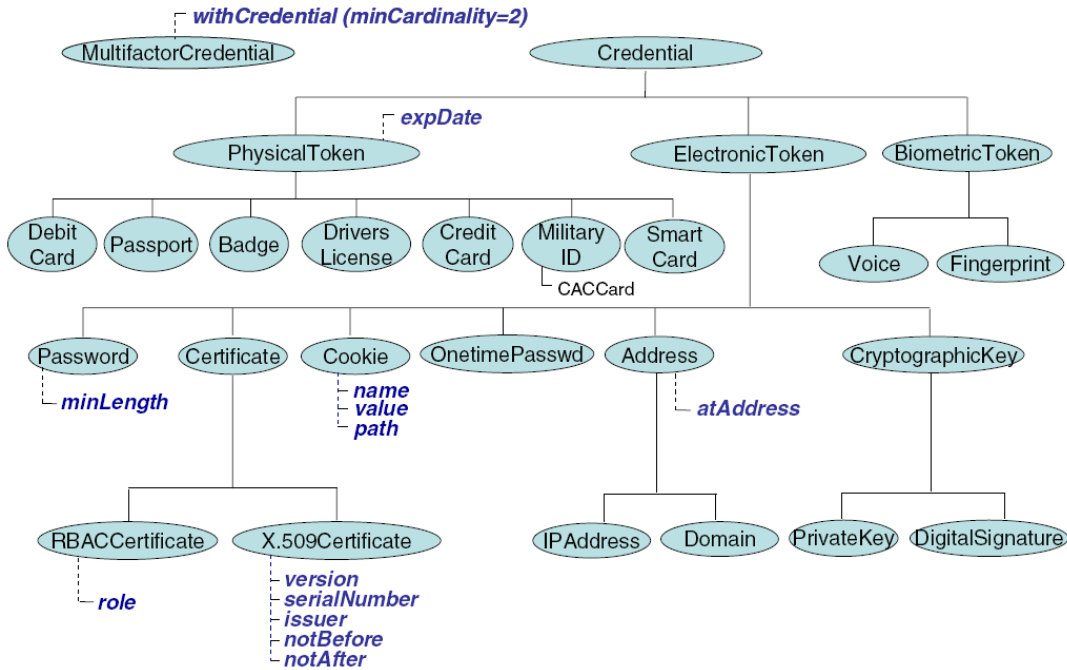


Figure 14 Credential Ontology [2]

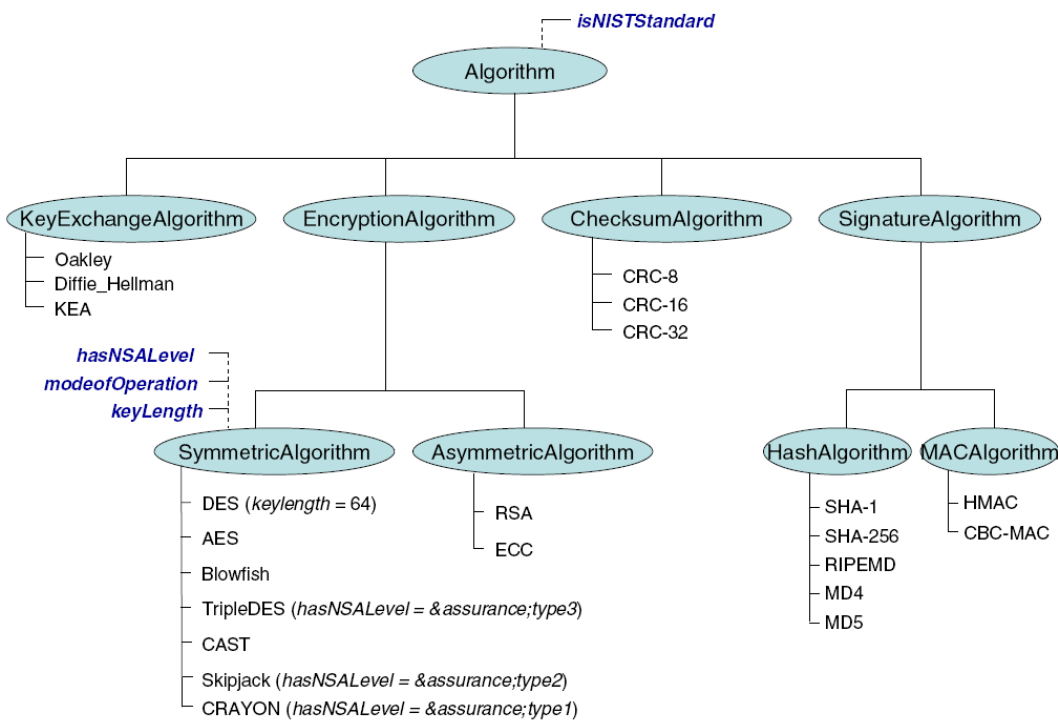


Figure 15 Security Algorithms Ontology [2]

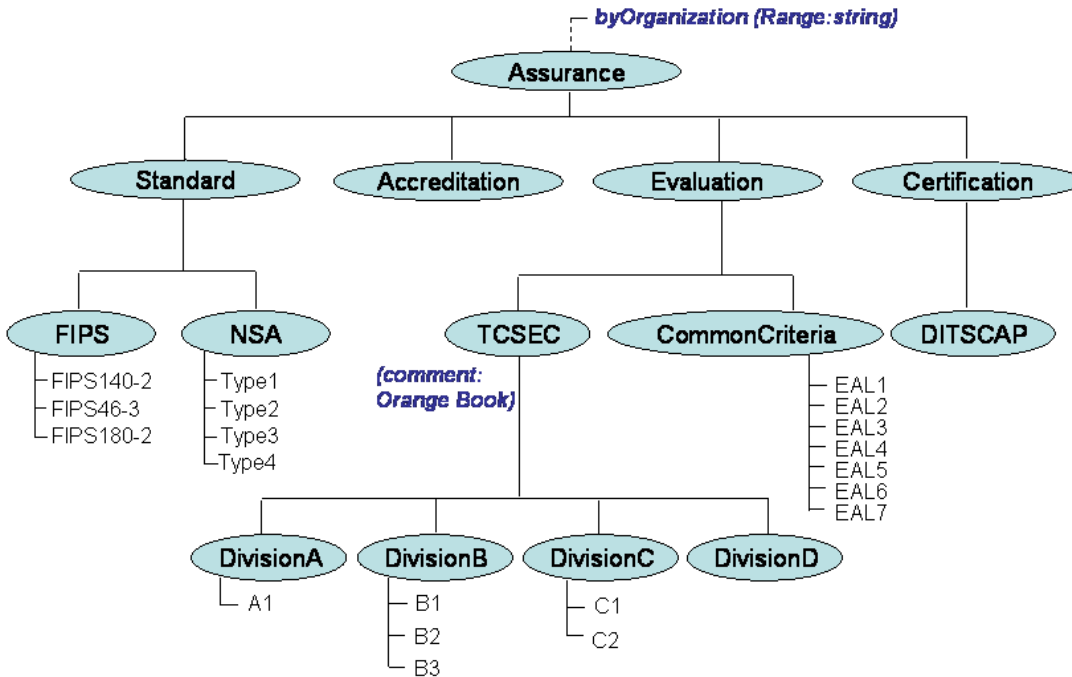


Figure 16 Security Assurance Ontology [1]

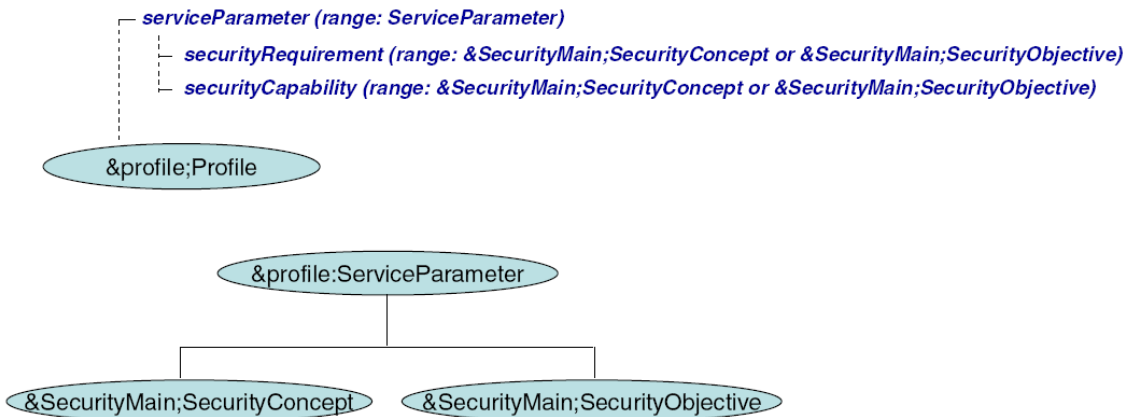


Figure 17 Service Security Ontology [2]



Figure 18 Agent Security Ontology [1]

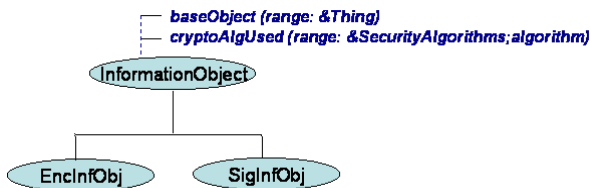


Figure 19 Information Object Ontology [1]

NRL security ontology is well organised concentrating mostly on security solutions area and providing a reasonable way to matchmaking between requirements and capabilities. When comparing NRL ontology to work by Savolainen et al. it can be noticed that the NRL security ontology lacks in security assets, threats and metric areas. However, NRL security ontology contains a substantially better description in ARTEMIS JU SP3 / 100017: Smart Objects For Intelligent Applications (SOFIA)

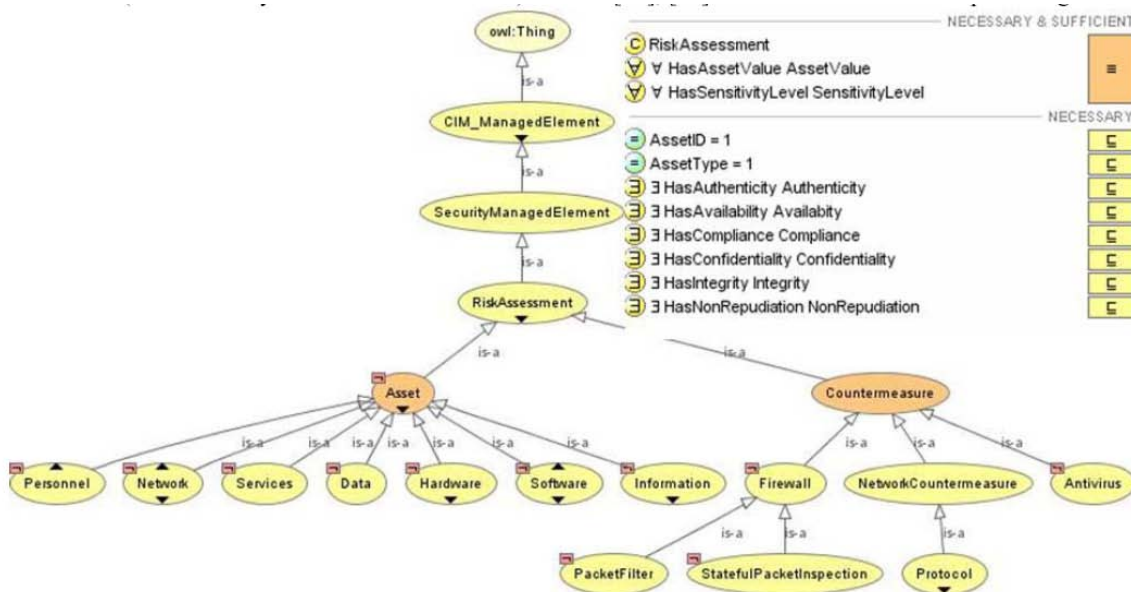


Figure 21 Part of Security Ontology in Protégé [8]

When compared ontology from Tsoumas et al. to ontology from Savolainen et al. following issues can be noticed. Firstly, both ontologies describe security in the same abstraction level, e.g. detailed protocols are not described. Secondly, Tsoumas et al. does not define security metrics for their ontology.

1.2.2. QoS Ontologies

QoS ontologies that are not directly intended for describing security are listed under this sub-section.

1.2.2.1. FIPA-QoS

FIPA-QoS ontology contains a basic vocabulary for the Quality of Service. Agents can utilise FIPA-QoS ontology when communicating about the Quality of Service. Thus, agents can query current QoS values from another agent, or alternatively, an agent can subscribe a notification when something happens related to the QoS. For instance, the agent can be informed when the throughput value of the sent real-time data dropped under the required QoS value[9].

1.2.2.2. DAML-QoS

DAML-QoS ontology constructed by Zhou et al.[10]complements DAML-S (nowadays OWL-S) ontology to take also service's non-functional aspects into account. DAML-QoS ontology contains three layers: 1) the QoS profile layer for matchmaking purposes, 2) the QoS property definition layer for elaborating the property's domain and range constrains, and 3) the metric layer for metrics definition and measurement. DAML-QoS makes web services machine understandable and facilitates especially service discovery. For example, the service enquiry returns services that may potentially satisfy the desired QoS requirements. DAML-QoS provides only integer values for QoS measurements, which are also used in the QoS measurement framework [11].

1.2.2.3. WSQoS-Ontology

Tian et al. [12] have defined WS-QoS approach for dynamic QoS-aware web-service selection and monitoring. One part of the WS-QoS approach is WS-QoSontology containing definitions for metric, protocol, and priority. However, this ontology is not stored in a standard ontology format (e.g. in OWL (Web Ontology Language) format) but is recorded in an XML format, and thus machine reasoning might be difficult. Nevertheless, the metric definition part specifies direction for each metric, e.g. higher is better, which is an essential feature to achieve the machine reasoning.

IOP principles		35 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

1.2.2.4. QoSOnt

Dobson et al. [13] describe a QoS ontology, called QoSOnt, for service-centric systems in order to enable communication about QoS between clients, providers, and intermediaries. The QoSOnt consists of three layers: 1) Base QoS and Units form the lowest layer, 2) Quality attributes constitute the middle layer, and 3) Usage Domains model the highest layer. Each layer is a separate ontology, and the third parties can replace these ontologies. The Base QoS ontology defines basic concepts common for all QoS that can be utilised in the higher layers. The Attributes layer specifies a QoS attribute e.g. for dependability. The dependability ontology may represent dependability attributes – reliability, availability, safety, and security – and a set of metrics for them.

1.2.2.5. Dependable Systems Ontology

Dependable Systems Ontology [14], also called ReSIST ontology – is an ontology about resilient and dependable systems. The ontology contains two main classes, i.e. Computer System Vulnerability and Dependable Systems Technology. The first class contains threats that can cause a computer system to malfunction or stop – and the latter class contains features and research areas of dependable systems technology. Mostly this ontology contains concepts related to dependability and security, but the metrics are not defined.

1.2.3. ONTOMETRIC

In reference [15] Lozano-Tello et al. present the ONTOMETRIC method for measuring suitability of the existing ontologies related to requirements of the intended usage. ONTOMETRIC method concentrates ontologies' characteristics from five dimensions, i.e. tools, language, content, methodology and costs. Authors listed characteristics from all of these dimensions and suggested useful grades for each characteristic, i.e. very-low, low, medium, high, very-high, or alternatively: non-supported / supported. Figure 22 is adopted from [15] – figure list characteristics related the content dimension[15].

DIMENSION: CONTENT	
CHARACTERISTIC	
CONCEPTS (FACTOR)	(very_low, low, medium, high, very_high)
Essential_Concepts	(very_low, low, medium, high, very_high)
Essential_Concepts_In_Superior_Levels	(very_low, low, medium, high, very_high)
Concepts_Properly_described_In_NL	(very_low, low, medium, high, very_high)
Formal_Specification_Of_Concepts_Coincides_With_NL	(very_low, low, medium, high, very_high)
Attributes_Describe_Concepts	(very_low, low, medium, high, very_high)
Number_Of_Concepts	(very_low, low, medium, high, very_high)
RELATIONS (FACTOR)	(very_low, low, medium, high, very_high)
Essential_Relations	(very_low, low, medium, high, very_high)
Relations_Relate_Appropriate_Concepts	(very_low, low, medium, high, very_high)
Formal_Specification_Of_Relations_Coincides_With_NL	(very_low, low, medium, high, very_high)
Arity_Specified	(very_low, low, medium, high, very_high)
Formal_Properties_Of_Relations	(very_low, low, medium, high, very_high)
Number_Of_Relations	(very_low, low, medium, high, very_high)
TAXONOMY (FACTOR)	(very_low, low, medium, high, very_high)
Several_Perspectives	(very_low, low, medium, high, very_high)
Appropriate_Not-Subclass-Of	(very_low, low, medium, high, very_high)
Appropriate_Exhaustive-partitions	(very_low, low, medium, high, very_high)
Appropriate_Disjoint-partitions	(very_low, low, medium, high, very_high)
Maximum_Depth	(very_low, low, medium, high, very_high)
Average_Of_Subclasses	(very_low, low, medium, high, very_high)
AXIOMS (FACTOR)	(very_low, low, medium, high, very_high)
Axioms_Solve_Queries	(very_low, low, medium, high, very_high)
Axioms_Infer_Knowledge	(very_low, low, medium, high, very_high)
Axioms_Verify_Consistency	(very_low, low, medium, high, very_high)
Axioms_Not_Linked_To_Concepts	(very_low, low, medium, high, very_high)
Number_Of_Axioms	(very_low, low, medium, high, very_high)

Figure 22 Characteristics related to the dimension content [15]

Utilising the ONTOMETRIC method as the wholeness might need lot of time, e.g. giving weights for each characteristic and calculating a grade for each dimension. However, method offers extensive set of characteristics that can be considered during the ontology selection process.

1.3. Overview of Security Design

1.3.1. Defining Security at Design-time

In SMEPP (Secure Middleware for Embedded P2P systems) project security requirements are defined by utilising scenarios [17]. Firstly, possible threats and their countermeasures, i.e. security mechanisms, are studied in [19]. After that, responsibilities related to possible threats are divided for the services, components and layers of the SMEPP middleware in [18]. In practice, scenarios are described as use-case diagrams, and after that possible security issues for each scenario are listed, which can occur during the described scenario.

Firesmith concentrates how to engineer security requirements in [20]. Firstly, he lists 12 classes of security requirements, which are as follows:

1. Identification requirements
2. Authentication requirements
3. Authorization requirements
4. Immunity requirements
5. Integrity requirements
6. Intrusion detection requirements
7. Nonrepudiation requirements
8. Privacy requirements
9. Security auditing requirements

10. Survivability requirements
11. Physical protection requirements
12. System maintenance security requirements

Secondly, he gives examples and guidelines to defining appropriate security requirements for each class. The means for defining quality attributes at design-time is represented by Niemelä et al. in [4]– and Figure 23 conforms this approach. However, figure is specialised for defining security. In this phase, ontologies work as an input for the requirements definition activity. Thus, ontologies described in the previous section have to be combined appropriately – in order to achieve one extensive ontology. This combined security ontology has to cover: assets, threats, security objectives, countermeasures (i.e. security mechanisms), metrics and connections between them. After that, the security ontology can facilitate a requirements definition activity comprehensively. Since, requirements will be derived from scenario descriptions in SOFIA project the security ontology acts as a guideline for this activity – in order to take security issues into account during the requirements definition phase.

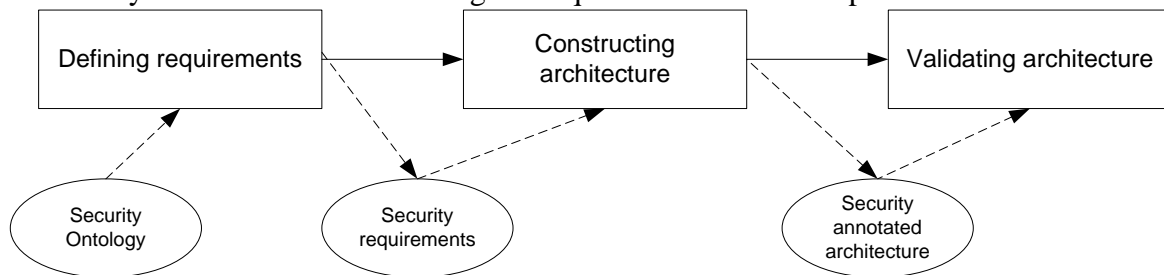


Figure 23 Overview of defining security at design-time

1.3.2. Defining Security at Run-time

Figure 24 presents an overview of the security management at the run-time. Firstly, available information and services are discovered from a smart space.

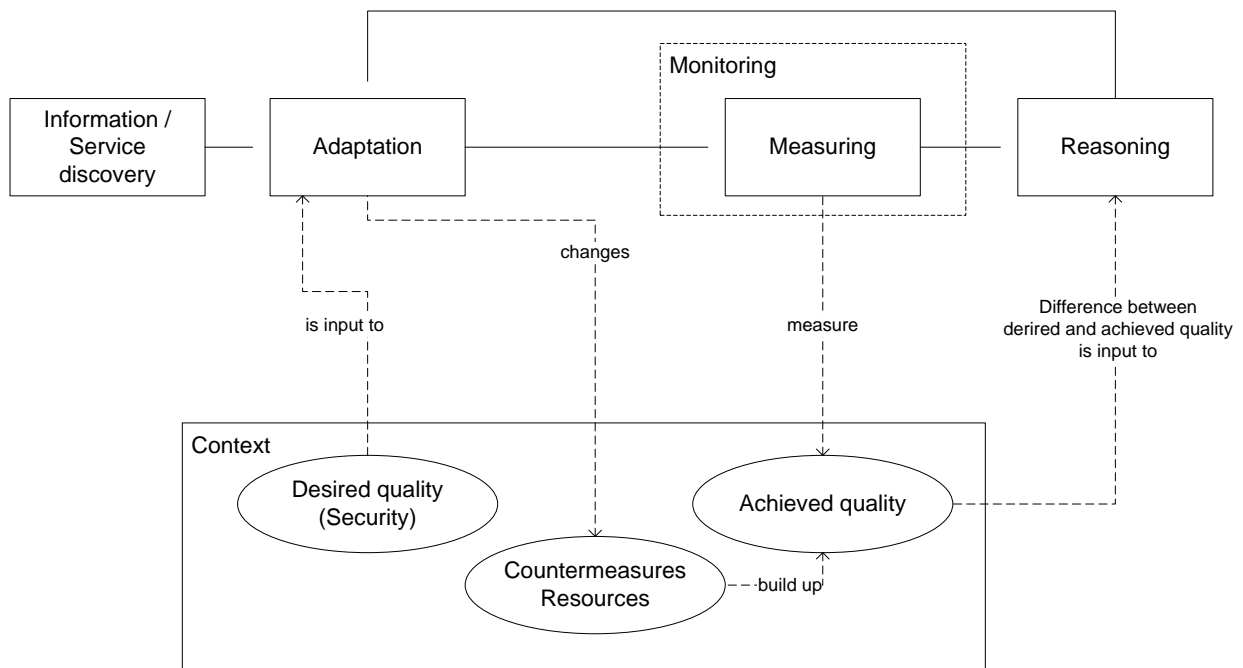


Figure 24 Run-time security (quality) management

. After that, adaptation is performed based on available information / services and a desired quality level. Thus, adaptation selects the most suitable countermeasure techniques (i.e. security solutions), resources etc. in order to achieve the desired quality level. Next, an achieved quality level is monitored by means of measuring. A measuring activity can measure several properties of the system, and from these values a

monitoring activity combines an overall security level. The monitoring activity reveals the achieved quality level – this value acts as an input for a reasoning activity. The reasoning activity calls the adaptation if the achieved quality level is not satisfying the desired quality level. In Figure 24 desired and achieved quality levels are included in a context because it affects to both quality levels.

In [3] Savolainen et al. presented a taxonomy of information security for service centric systems. This taxonomy has a metrics class which contains few security metrics collected mostly from [3]– Table 2 lists run-time applicable metrics from these.

Metric	Description	Formula
Access auditability	How complete is the audit trail concerning the user access to the system and data?	$X=A/B$ A= number of user accesses to the system and data recorded in the access history database B=number of user accesses to the system and data done during evaluation. 1.0 is the optimal value
Access controllability	How controllable is access to the system?	$X=A/B$ A=number of detected different types illegal operations B= number of types of illegal operation as in the specification 1.0 is the optimal value
Network vulnerability	How vulnerable is the system?	$X = A / B$ A =number of detected different types intrusions B =number of types of vulnerabilities specifications 0 is the optimal value $Y = C / T$ C = number of detected intrusions T = a period of operation time (run-time)
Attack frequency	How many attacks have been detected in a specific time frame?	$X=A/T$ A= number of detected attacks T= period of operation time
Data corruption prevention	What is the frequency of data corruption events?	$X = 1 - A / N$ A = number of times that a major data corruption event occurred N = number of test cases tried to cause data corruption event $Y = 1 - B / N$ B=number of times that a minor data corruption event occurred $Z = A / T$ and B / T ; T = period of operation time (during operation testing) (run-time)

Table 2 Run-time metrics

It is clear that above mentioned list of metrics is not enough for covering the whole security area (cf. classes of security requirements in section 4.2). Thus, it is necessary to find a suitable way to measure security more extensively. Wang & Wulf presented a security measurement framework in [21]. The main idea in their work is to divide security to smaller parts, i.e. decomposition, and finally these smallest parts can be measured. Figure 25 shows the decomposition made for non-repudiation in [21]– attached to base measures and derived measures terms from [22].

IOP principles		39 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

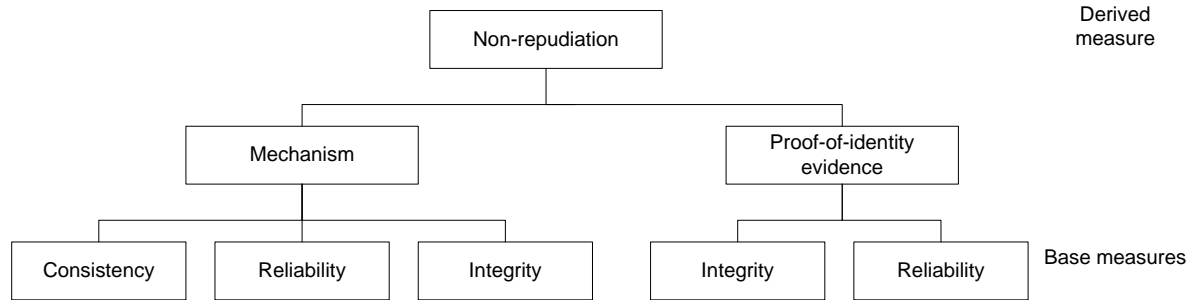


Figure 25 Decomposition of non-repudiation

Therefore, covering all security requirements classes presented in section 4.2 demands to perform a decomposition task for each class and selecting a suitable metric for each leaf node. Based on the work of Wang and Wulf root level's security can be calculated by utilising values from leaf nodes.

1.3.3. References

1. NRL Security Ontology, URL: <http://chacs.nrl.navy.mil/projects/4SEA/ontology.html>
2. Anya Kim, Jim Luo, and Myong Kang, Security Ontology for Annotating Resources, OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, Springer Verlag, pp. 1483-1499
3. Savolainen Pekka, Niemelä Eila and Savola Reijo, A Taxonomy of Information Security for Service-Centric Systems, in 33rd EUROMICRO Conference on Software Engineering and Advanced Applications 2007, pp. 5-12.
4. Niemelä E, Evesti A. and Savolainen P, Modeling Quality Attribute Variability, in 3rd international conference on Evaluation of Novel Approaches to Software Engineering (ENASE), 2008, pp. 169-176.
5. Denker G, Kagal L, Finin T, Paulucci M and Sycara K, Security for DAML web services: Annotation and matchmaking, Book chapter in The Semantic Web – ISWC 2003, p. 335-350.
6. Denger G, Kagal L., Finin T. Security in the Semantic Web Using OWL, Information Security Technical Report, 2005, Vol. 10 No: 1, pp. 51-58
7. Tsoumas B., Dritsas S. and Gritzalis D. An Ontology-Based Approach to Information Systems Security Management, Book chapter in Computer Network Security, Vol. 3685/2005, pp. 151-164.
8. Tsoumas B. and Gritzalis D. Towards an Ontology-Based Security Management, in 20th International Conference on Advanced Information Networking and Applications (AINA), 2006, pp. 985-992.
9. Foundation for Intelligent Physical Agents, FIPA Quality of Service Ontology Specification, URL: <http://www.fipa.org/specs/fipa00094/XC00094.html>
10. Zhou C., Chia L.-T. and Lee B.-S. DAML-QoS Ontology for Web Services, in IEEE International Conference on Web Services, 2004, pp. 472-479.
11. Zhou C., Chia L.-T. and Lee B.-S. QoS Measurement Issues with DAML-QoS Ontology, in IEEE International Conference on e-Business Engineering, 2005, pp. 395-402.
12. Tian M., Gramm A., Ritter H. and Schiller J., Efficient Selection and Monitoring of QoS-Aware Web Services with the WS-QoS Framework, in IEEE/WIC/ACM International Conference on Web Intelligence, 2004, pp. 152-158.
13. Dobson G., Lock R. and Sommerville I., QoSOnt: a QoS ontology for service-centric systems, 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 2005, pp. 80-87.

IOP principles		40 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

14. Dependable Systems Ontology URL: <http://users.ecs.soton.ac.uk/aoj04r/resist.owl>
15. Lozano-Tello A and Gómez-Pérez A., ONTOMETRIC: A Method to Choose the Appropriate Ontology, in Journal of Database Management, 2004, Vol. 15 No: 2, pp 1-18.
16. W3C, OWL Web Ontology Language Overview, W3C Recommendation 10 February 2004, URL: <http://www.w3.org/TR/owl-features/>
17. SMEPP project, URL: <http://www.smepp.org/>
18. SMEPP, Deliverable D3.2 Conceptual Architecture of Secure EP2P Middleware URL: http://www.smepp.org/DownloadsReview/D3_2_Final_Year2.pdf
19. SMEPP, Deliverable D4.1 Threat Models for EP2P Networks, URL: http://www.smepp.org/DownloadsReview/D4_1_Final_Year1.pdf
20. Firesmith D. Engineering Security Requirements, in Journal of Object Technology, 2003, Vol. 2 No: 1, pp. 53-68
21. Wang C. and Wulf W. A., A Framework for Security Measurement, Proceedings of the National Information Systems Security Conference, Baltimore, pp. 522-533, 1997 (available in URL: <http://csrc.nist.gov/nissc/1997/proceedings/522.pdf>)
22. Garcia F., Bertoa M. F., Calero C., Vallecillo A., Ruíz F. and Genero M. Towards a consistent terminology for software measurement, in Information and Software Technology, 2006, Vol 48 No: 8 , pp. 631-644
23. ISO/IEC 9126 parts 1-4, Software engineering – Product quality.

IOP principles		41 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

1.4. Annex 1: Rapid Overview of Policies for Networks, Systems and Service Management

Policies, which constrain the behavior of system components, are becoming an increasingly popular approach to simplify the management and dynamic adjustability of applications in academia and industry. There are many benefits deriving from the adoption of policy-based approaches, including reusability, efficiency, extensibility, context-sensitivity, verifiability, support for both simple and sophisticated components, protection from poorly-designed, buggy, or malicious components, and reasoning about component behavior. Policies have important analogues in animal societies and human cultures [1].

Policy-based network management has been the subject of extensive research over the last decade [1]. Policies are often applied to automate network administration tasks, such as configuration, security, recovery, or quality of service (QoS). In the network management field, policies are expressed as sets of rules governing choices in the behavior of the network.

The Internet Engineering Task Force (IETF) has proposed an Internet-draft of terminology for describing network policy [2] and provides many definitions. In particular, a policy is formally defined as an aggregation of policy rules. Each policy rule is made up of a set of conditions and a corresponding set of actions. The conditions define when the policy rule is applicable. Once a policy rule is so activated, one or more actions contained by that policy rule may then be executed. These actions are associated with either meeting or not meeting the set of conditions specified in the policy rule. In other words, a policy specifies what action(s) must be taken when a set of associated conditions are met. Network policies are grouped into three general areas: how the policy is used, how the policy is triggered, and at which level the policy is applied. Policies can be categorized into different types according to their purpose and intent. For instance, configuration policies define the generic setup of a managed entity, security policies deal with verifying that a client is actually who the client claims to be, permitting or denying access to resources, selecting or applying authentication mechanisms, and performing accounting and auditing of resources.

Multiple approaches for policy specification have been proposed that range from formal policy languages that can be processed and interpreted easily and directly by a computer, to rule-based policy notation using an if-then-else format, to the representation of policies as entries in a table consisting of multiple attributes [3, 4, 5]. In particular, there has been a great deal of research on the topic of formal representation of policies. Much of this research has been in the area of representing security policies, and the more general problem of translating ambiguous natural language policies into some type of formal representation. Representing network policies with an unambiguous language is the key to detecting conflicts. Once the policies are in a logical representation, methods already developed from research in this area will provide a means to check the consistency of multiple policies. Recently, the trend in the policy specification research area is the adoption of semantically-rich policy representations to reduce human error, simplify policy analysis, reduce policy conflicts, and facilitate interoperability. For instance, the importance of adopting a high level of abstraction for the specification of all security policy building elements (subjects, actions, context, etc..) is starting to emerge in well-known policy frameworks, such as KAoS and Rei [6, 7].

There are also ongoing standardization efforts toward common policy information models and frameworks. The Internet Engineering Task Force, for instance, has been investigating policies as a means for managing IP-multiservice networks by focusing on the specification of protocols and object-oriented models for representing policies. The IETF do not define a specific language to express network policies but rather a generic object-oriented information model for representing policy information following a rule-based approach. In particular, the IETF tries to standardize a

IOP principles		42 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

scalable architecture for policy administration and distribution of network policies. The architecture comprises several modules to accomplish certain management tasks: creating rules/policies, checking for policy conflicts, storing/distributing policies, converting policies into commands that network devices can interpret and understand. An underlying assumption of this draft is that policies are stored in a centralized repository. The policy repository is one of three important entities of the model. The other entities are the policy enforcement points (PEPs) and policy decision point (PDP). A PEP is a component of a network node (e.g., a router, switch, or hub) where the policy decisions are actually enforced. When the PEP requires a policy decision about a new flow of traffic or authentication, for example, the PEP will send a request to a PDP. The PDP is the entity in the network where policy decisions are made. This PDP, which may reside on a remote server, will make policy decisions using information retrieved from policy repositories. Communication is needed to and from the policy repository as well as between the PDP and the PEP. In many proposals the policy repository is a directory, and therefore the appropriate access protocol would be the Lightweight Directory Access Protocol (LDAP). Examples of a policy protocol, which is used to request and reply to policy decisions, could be the Common Open Policy Service (COPS) protocol and Simple Network Management Protocol (SNMP).

1.4.1. References

- [1] S. Wright, R. Chadha, G. Lapiotis (eds.): Special Issue on Policy Based Networking. IEEE Network, Vol. 16, No. 2, March, (2002), 8-56
- [2] The IETF Policy Framework Working Group: Charter available at <http://www.ietf.org/html.charters/policy-charter.html>
- [3] J. Fritz Barnes, R. Pandey: CacheL: Language Support for Customizable Caching Policies. In Proceedings of 4th International Web Caching Workshop, San Diego, CA, (1999)
- [4] J. Hoagland: Specifying and Implementing Security Policies Using LaSCO, the Language for Security Constraints on Objects. Ph.D. dissertation, UC Davis, (2000)
- [5] N. Damianou, N. Dulay, E. C. Lupu, M. Sloman: Ponder: A Language for Specifying Security and Management Policies for Distributed Systems. Imperial College, UK, Research Report Department of Computing 2001, (2000)
- [6] J. M. Bradshaw, A. Uszok, R. Jeffers, N. Suri, M. Burstein: Representation and reasoning for DAML-based policy and domain services in KAoS and Nomads. Proceedings of the Autonomous Agents and Multi-Agent Systems Conference (AAMAS 2003). Melbourne, Australia, New York, NY: ACM Press, pp. 835-842, (2003)
- [7] L. Kagal, T. Finin, A. Johshi: A Policy Language for Pervasive Computing Environment. In Proceedings of IEEE Fourth International Workshop on Policy (Policy 2003). Lake Como, Italy, 4-6 June, Los Alamitos, CA: IEEE Computer Society, pp. 63-76, (2003).

IOP principles		43 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

2. IOP Requirements From Vertical Scenarios

The purpose of this section is to define the IOP requirements distilled from the scenarios defined for the personal spaces (WP1), smart indoor spaces (WP2) and smart cities (WP3). Thereafter the defined requirements have been prioritized based on the assessment criteria defined by the project management team.

The motivation behind the IOP requirements specification and analysis of the vertical scenarios is to derive appropriate principles for the development of the *SOFIA InterOperability Platform (IOP)*.

The resulting IOP should not only ensure interoperability at both information and service level, but also speed up and simplify the design, development, and deployment of cross-domain applications based on smart spaces.

Each of three partners active in T5.1 (i.e., VTT, UNBO and UNRO) has been responsible for analyzing all the scenarios provided by one vertical WP:

- VTT for use case scenarios from WP1
- UNIBO for use case scenarios from WP2
- UNIROMA for use case scenarios from WP3

After scenario analysis the IOP requirements were defined and ranked based on the amount of scenarios they were related to. Thereafter, business analysis was made for all scenarios defined by vertical work packages. As a result of the second analysis the scenarios with the highest business impact and lowest risk to implement were identified in each application domain. The following scenarios got the highest priority:

- WP1: Enhanced navigation scenario and Mobile use preference
- WP2: Support of maintenance operators on site and a set of sub-scenarios that can be exploited in its realization.
- WP3: Outage triggered operation, Dining and buying shoes and, Subway station.

As a conclusion, this section presents two lists of IOP requirements of the highest priority, one related to quality requirements and another related to functional/non-quality requirements. The template to describe the requirements to be satisfied by the IOP is specified in Par. 5.1.

The requirements are used as driving factors in the conceptualization and in the definition of the principles of the IOP; principles guide design and implementation; evaluation criteria define the minimum set of criteria that each IOP has to fulfill.

2.1. Requirements

This section provides a list of IOP requirements. We tried to keep low the total number of requirements, to avoid lengthy and too detailed descriptions. To achieve the purpose of a limited set of requirements, the most frequent requirements have been identified and similar aspects deriving from different scenarios have been merged together, from both the same vertical (intra-vertical analysis) and different verticals (inter-vertical analysis).

2.1.1. Quality Requirements

Security

Definition:

This category of requirements defines the required sub-characteristics (Confidentiality, Integrity, Privacy and Non-Repudiability) of security and the related countermeasures.

ID - Type	5.1 – Q
Name:	User/device/smart space application authentication
Description:	IOP must allow user and device authentication with different security solutions (e.g. id/password, public key exchange, biometrics, etc.). Authentication could be required before allowing different levels of interaction and different types of operations.
Rationale:	If user preferences stored in a mobile phone provide an access to a smart space (e.g. smart home) the user of the phone must be authenticated in a proper way. Similarly, before information exchange, application and smart space have to be authenticated, possibly depending on the sensitivity of exchanged information. Access to reserved services offered in a smart space should be granted only to authorized users/services/devices.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: high
Space Significance:	Priority for WP1...WP3: high
Relation to Interoperability:	As simple as possible in order to deal variations
Related scenarios:	User: WP_1.UC_VTT5-Feb-6 Application: WP_1.UC_NOK1-Feb-6 WP_1.UC_NOK2-Feb-6 WP_1.UC_VTT5-Feb-6 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S3/4
Existing enablers:	Credential ontologies, existing authentication mechanisms
Required enablers:	Dynamic activation/selection of countermeasures

IOP principles	45 (98)
D5.11-v1.0	Confidentiality Level: PU (Public) 2010-01-09

ID - Type	5.2 – Q
Name:	Access control to smart appliances and related authorization
Description:	<p>When a user/service tries to access any device/smart appliance (e.g., light adjustment in a smart home), IOP should check that it is allowed to access that appliance, based on its (administrative or network) domain and/or other relevant conditions, e.g. location or other contextual conditions.</p> <p>This requirement also regards any software update installation, which should not breach access control.</p> <p>Enforcing access control might require support for authorization.</p>
Rationale:	<p>Access control is needed to prevent unauthorized usage of smart space appliances by services that do not have proper credentials to do so.</p> <p>This is important to ensure the correct behavior of smart home appliances and to keep automated appliances under control.</p>
Architectural Significance:	Priority for WP5: high (access control solutions should be considered during architectural design)
Application Significance:	Priority for WP6: medium (applications can rely on the existence of access control support)
Space Significance:	Priority for WP2, WP3: high
Relation to Interoperability:	Information and service interoperability
Related scenarios:	<p>Samples from personal spaces scenarios:</p> <p>WP_1.UC_NOK1-Feb-6 WP_1.UC_NOK2-Feb-6 WP_1.UC_VTT5-Feb-6</p> <p>Samples from smart environment scenarios</p> <p>WP2.UC1.0-160209-S_1, 2, 3, 4, 5, 6, 7 WP2.UC2.0-160209-S_1 WP2_UC1_S1,2,3,4 WP2-Smart Lighting.1-16-2-2009-S1,2,3 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S3/4</p>
Existing enablers:	Authorization lists, / group (e.g. only for family members) / role based (e.g. only for caretakers) authorization
Required enablers:	Access control policy-based system, support for collecting context information

ID - Type	5.3 – Q
Name:	Shared information integrity and privacy
Description:	<p>Information about both user preferences/profiles and smart space appliances (e.g., coming from sensors or other devices) should be protected during transmission between IOP, services, and sensors.</p> <p>IOP has to prevent unauthorized corruption of e-mails, their attachments, SMS messages, and, more generally, any kind of transmitted information (integrity and sometimes prevention from forging).</p> <p>In some cases, there are relevant privacy requirements to consider: personal information about users (preference, personal information, sensible data, etc.) must not be stored in association with other information.</p>
Rationale:	<p>Some service might distribute malicious data or service by accident or on purpose, and this distribution has to be recognized and prevented.</p> <p>It is also important to guarantee that message exchanges between users and smart appliances is protected, in order to avoid altering sensitive information.</p> <p>User information and sensor-provided information about the smart home might be privacy-related; therefore they should not be disclosed to third parties.</p>
Architectural Significance:	Priority for WP5: high
Application Significance:	Priority for WP6: medium
Space Significance:	Priority for WP2: high Priority for WP3: medium
Relation to Interoperability:	Information interoperability
Related scenarios:	<p>WP_1.UC_NOK1-Feb-6</p> <p>WP_1.UC_NOK2-Feb-6</p> <p>WP_1.UC_NOK3-Feb-6</p> <p>WP_1.UC_NOK4-Feb-6</p> <p>WP_1.UC_NOK5-Feb-6 (message integrity)</p> <p>WP_1.UC_NXP3-Feb-6 (message integrity)</p> <p>WP_1.UC_NXP4-Feb-6 (message integrity)</p> <p>WP_1.UC_VTT1-Feb-6</p> <p>WP2_UC1_S1,2,3,4</p> <p>WP2-Smart Lighting.1-16-2-2009-S1,2,3</p> <p>WP2_CAC1</p> <p>WP2.UC1.0-160209-S_1, 2, 3, 4, 5, 6, 7</p> <p>WP2.UC2.0-160209-S_1</p> <p>WP2.GS1.1-090213-GS1, 2, 3, 4</p> <p>WP2-Smart Lighting.1-16-2-2009-S1,2,3</p> <p>WP_3.UC1_S1</p>

IOP principles		47 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

	WP_3.UC2_S1/2/3a/3b WP_3.UC3_S3/4
Existing enablers:	N/A
Required enablers:	Support for encryption

ID - Type	5.4 – Q
Name:	Non-repudiability of performed operations and requests
Description:	<p>Every action performed (e.g. execution of a service) by a user or device must be logged and univocally associated to the source of the action.</p> <p>For instance, whenever a request for home maintenance, a maintenance operation or light adjustment operation is performed, IOP should explicitly account for this action and associate it to the subject (user, maintenance operator) who has performed it.</p>
Rationale:	<p>To avoid possible legal/operational issues, it is necessary to ensure that this action cannot be denied by the subject (user, maintenance operator) who has performed it.</p> <p>In case of home maintenance, legal issues might arise. In case of conflicting needs about light adjustment, non-repudiability is needed to allow conflict resolution and tracking.</p> <p>However, note that these cannot breach privacy requirements.</p>
Architectural Significance:	Priority for WP5: high
Application Significance:	Priority for WP6: high
Space Significance:	Priority for WP2, WP3: high
Relation to Interoperability:	Information interoperability
Related scenarios:	WP2.UC1.0-160209-S_6, 7 WP2.GS1.1-090213-GS1, 2, 3 WP2_UC1_S4 WP2-Smart Lighting.1-16-2-2009-S3 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b
Existing enablers:	Timestamps and authenticated identity information
Required enablers:	Digital signature, authentication modules

ID - Type	5.5 – Q
Name:	User/smart space/service immunity
Description:	User/smart spaces should protect themselves from infections. Harmful contents are not used or passed forward (to another application/service or smart space).
Rationale:	Some service might distribute malicious data or service by accident or on purpose, and this distribution has to be recognized and prevented.
Architectural Significance:	Priority for WP5: high
Application Significance:	Priority for WP6: medium
Space Significance:	Priority for: WP3 high
Relation to Interoperability:	Information interoperability
Related scenarios:	WP_1.UC_NOK5-Feb-6 (message integrity) WP_1.UC_NXP4-Feb-6 (message integrity) WP_1.UC_NXP3-Feb-6 (message integrity) WP2.GS1.1-090213-GS1, 2, 3 (both) WP2_UC1_S1,2,3,4,5 WP2-Smart Lighting.1-16-2-2009-S1,2,3 WP2_CAC1 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S3/4
Existing enablers:	Antivirus programs, firewalls, cryptography, hash codes.
Required enablers:	Security level recognition of established connection

ID - Type	5.6 – Q
Name:	Security monitoring (auditing)
Description:	Used security mechanisms and the achieved security level of application / service and smart space have to be audited (monitored at a real-time)
Rationale:	Auditing ensures that the required security level is achieved and a user is aware of changes on the security level. In other words, this informs if above mentioned requirements cannot be fulfilled.
Architectural Significance:	Priority for WP5: high
Application Significance:	Priority for WP6: low
Space Significance:	Priority for WP1: medium Priority for WP3: high
Relation to Interoperability:	Information and service levels
Related scenarios:	WP_1.UC_NOK1-Feb-6, WP_1.UC_NOK2-Feb-6, WP_1.UC_NOK5-Feb-6,

IOP principles		49 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

	WP_1.UC_VTT5-Feb-6, WP_1.UC_NXP4-Feb-6, WP_1.UC_NXP3-Feb-6, WP2.GS1.1-090213-GS1, 2, 3, 4 WP2_UC1_S1,2,3,4 WP2-Smart Lighting.1-16-2-2009-S1,2,3 WP2_CAC1 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b
Existing enablers:	Security ontologies, different countermeasure techniques
Required enablers:	Monitoring mechanisms (metrics etc.)

ID - Type	5.7 – Q
Name:	Intrusion detection (also related to 5.2 - F)
Description:	In spaces like, home, personal car, restricted areas in public infrastructures etc. All unauthorized attempts to access data have to be prevented and informed to the space/content owner.
Rationale:	If someone (with his/her device) tries to continuously get content that is not authorized for it, these attempts are informed for an appropriate person, and if needed device is blocked.
Architectural Significance:	Priority for WP5: high
Application Significance:	Priority for WP6: medium
Space Significance:	Priority for WP1: medium Priority for WP3: high
Relation to Interoperability:	Information / service levels
Related scenarios:	WP_1.UC_NOK1-Feb-6 WP_1.UC_NOK2-Feb-6 WP_1.UC_VTT5-Feb-6 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b
Existing enablers:	Alarms, event logs
Required enablers:	N/A

IOP principles		50 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Availability

Definition:

This section encompasses availability of services and survivability of IOP.

ID - Type	5.8 – Q
Name:	Availability/survivability of IOP, services, and resources
Description:	<p>Fundamental services should be always available despite possible failures in single devices/sensors/services.</p> <p>IOP must tolerate multiple losses of resources and continue to function properly (or with reduced quality) also when disasters produce failures in a large part of the system.</p> <p>IOP shall keep records on capabilities of available services and resources and it will make it visible when a new service/resource is available.</p>
Rationale:	<p>For instance, in emergency situation information has to be available for a rescue team even if part of systems are unavailable.</p> <p>In general, when a user joins a smart space she/he must be able to use those services/resources (i.e. devices) that are available and fit the best to her/his needs.</p>
Architectural Significance:	Priority for WP5: high
Application Significance:	Priority for WP6: medium
Space Significance:	Priority for WP1: medium Priority for WP3: high
Relation to Interoperability:	Information interoperability (state/control/user preferences/location/) and service interoperability (alternative services/resources, their capabilities and status of capabilities (e.g. battery); history data of used services)
Related scenarios:	<p>WP1_UC_NOK1-Feb-6</p> <p>WP1-UC_NOK2-Feb-6</p> <p>WP1-UC_NOK3_Feb-6</p> <p>WP2.GS1.1-090213-GS4</p> <p>WP2_UC1_S4</p> <p>WP2-Smart Lighting.1-16-2-2009-S3</p> <p>WP_3.UC1_S1</p> <p>WP_3.UC2_S1/2/3a/3b</p> <p>WP_3.UC3_S1/2/3/4/5</p>
Existing enablers:	Service discovery, resource mgmt service
Required enablers:	Event service, dynamic service discovery

Performance

Definition:

The performance category includes relevant performance indicators such as response time and scalability.

ID - Type	5.9 – Q
Name:	Real-time notification and information delivery (with controlled response time)
Description:	<p>IOP shall be able to deliver fast and real-time information:</p> <ul style="list-style-type: none"> - alarms and notifications; outage, motion and security alerts, - real-time video streams, pictures, maps, information about emergency exits, localization information, routing, ... - reconfiguration events, evacuation/operation directives - damage reports (location, images, status, etc.) - sensor faults - information about user activity to adjust home appliances (e.g., light) light accordingly
Rationale:	<p>Hard performance and reliability requirements at the same time: Fast alarms and video-streaming with safety critical information (i.e. medical info, emergency exits, safe locations). Timely info about sensor faults is needed to perform maintenance operations. Notification about user activity is needed to automatically adjust smart home appliances.</p>
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: high
Space Significance:	Priority for WP3: high
Relation to Interoperability:	Information interoperability (state/control/location) and service interoperability (ad hoc networking, remote medical service, real-time video streaming)
Related scenarios:	<p>WP2.GS1.1-090213-GS1, 2, 3 WP2.UC1.0-160209-S_1, 2, 3, 4, 5, 6, 7 WP2.UC2.0-160209-S_1 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b</p>
Existing enablers:	Service discovery, resource mgmt service
Required enablers:	Event service, dynamic service discovery, situation based performance and reliability mgmt (i.e. adaptation for survivability)

IOP principles		52 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

ID - Type	5.10 – Q
Name:	Scalability of IOP
Description:	<p>IOP shall be able to deliver the expected level of performance regardless of:</p> <ul style="list-style-type: none"> • the scaling amount of sensors and video cameras • the scaling number of end-users (big stadium vs. small cinema) • the scaling amount of resources (e.g. memory and CPU power) available on a SOFIA device (i.e. a device with the ability to access SOFIA information world)
Rationale:	<p>The amount of information producers and consumers is not known beforehand in emergency situations and therefore, IOP has to scale according to the number of information producers and consumers. Similarly, the amount of sensors/devices managed by the maintenance company of a building generally depends on the building size and on the available technological equipment.</p> <p>The IOP should also allow implementation scalability so that the IOP can be implemented on resource rich devices e.g. a laptop computer but also on very resource constrained devices such as a lamp. A computer has a big CPU and lots of memory while a lamp would have a limited memory and a small CPU.</p>
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: high
Space Significance:	Priority for WP3: high
Relation to Interoperability:	Information interoperability (state/control/location) and service interoperability (ad hoc networking, remote medical service, real-time video streaming)
Related scenarios:	<p>WP2.GS1.1-090213-GS1, 2, 3</p> <p>WP2-Smart Lighting.1-16-2-2009 S1/S2/S3</p> <p>WP_3.UC1_S1</p> <p>WP_3.UC2_S1/2/3a/3b</p> <p>WP_3.UC3_S1/2/4/5</p>
Existing enablers:	Service/resource discovery
Required enablers:	Resources optimization; duplication of services, shared memory bypassing

IOP principles		53 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Usability

ID - Type	5.11 – Q
Name:	Usability of end-user services (uniform interface and presentation of semantic connections)
Description:	IOP shall help in making end-user services which are self-explanatory, nonintrusive, easy-to-learn and easy-to-use. Also semantic connections with other services and resources should be explicitly shown to the final users.
Rationale:	When the user joins a smart space she/he is provided (information) services that can be easily taken into use.
Architectural Significance:	Priority for WP5; low
Application Significance:	Priority for WP6: high
Space Significance:	Priority for WP1-3: high
Relation to Interoperability:	Device, service and information interoperability
Related scenarios:	WP2.GS1.1-090213-GS1, 2, 3, 4 WP2_UC1_S1,2,3,4 WP2-Smart Lighting.1-16-2-2009-S1,2,3 WP2_CAC1 WP2_TUeID_SC_S_1 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S1/2/3/4/5
Existing enablers:	N/A
Required enablers:	Adaptation of a service/information according to user's terminal Presentation of semantic connections

IOP principles		54 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Adaptability

Definition:

This class of requirements includes the needs of runtime and automatic support for adaptation to different kinds of smart spaces.

ID - Type	5.12 – Q
Name:	Context-based service adaptation
Description:	The service shall automatically adapt according to the user's situation, needs and available resources.
Rationale:	The user of the smart space shall be able to use the resource/service/information that best meets his/her requirements
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: high
Space Significance:	Priority for WP1: low Priority for WP2: high Priority for WP3: high
Relation to Interoperability:	Information and service levels Content (i.e. information) is adapted according to the used resources,
Related scenarios:	Several ones, e.g. WP1_UC_NOK1-Feb-6 WP2.UC1.0-(CAL)160209-S_1, 2, 3, 4, 5, 6, 7 WP2.UC2.0-160209-S_1 WP2_UC1(NOK)_S1,2,3,4 WP2-Smart Lighting.1-16-2-2009-S1,2,3 WP2_CAC1 WP2_(CCC)GS2 GS3 GS4 WP2_TUeID_SC_S_1 WP_3.UC3_S1/2/3/4/5
Existing enablers:	Several adaptation strategies and tools, e.g., genetic algorithms
Required enablers:	Interoperability at the protocol and access control levels (i.e. communication & connectivity) Service adaptation reasoning based on available resources and required QoS level. Location engine Generalized context detector

IOP principles		55 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

ID - Type	5.13 - Q
Name:	Automatic reconfiguration
Description:	The smart space shall be able to adapt its behavior in relation to specific events it has detected/is notified of.
Rationale:	The smart space interacts with the environment to automatically adapt it to best fit user requirements in a context-aware fashion, e.g., increasing room luminosity if a user switches from a music listening to a pdf reader application
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: low
Space Significance:	Priority for verticals: high
Relation to Interoperability:	Information / service level Environment adapted in relation to the used resources and services
Related scenarios:	WP2.UC1.0-160209 S_1, S_2, S_3, S_4, S_5 WP2.UC2.0-160209 S_1 WP2_UC1_S1,2,3,4 WP2-Smart Lighting.1-16-2-2009-S1,2,3 WP2_CAC1 WP2_SC_S_1 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b
Existing enablers:	Monitoring and quality measuring techniques
Required enablers:	Interoperability at the protocol and access control levels (i.e. communication & connectivity)

Integrability

Definition:

Integrability requirements consider the need for the artifacts to be easily exploited and utilized in different smart spaces, even after dynamic discovery and integration.

ID - Type	5.14 – Q
Name:	Heterogeneous interfaces and implementation languages
Description:	Diverse applications/services must be able to communicate with each other, despite of heterogeneous programming interfaces and/or implementation languages.
Rationale:	In smart space/smart home environment, applications/services are developed by different stakeholders who choose their own implementation technologies. IOP must be able to cope with such diversity.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: high
Space Significance:	Priority for WP1-3 high
Relation to Interoperability:	Information and service interoperability
Related scenarios:	Many, e.g. WP_1_UC_NX2-Feb-6 WP2.GS1.1-090213-GS1, 2, 3, 4 WP2.UC1.0-160209-S_1, 2, 3, 4, 5 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S1/2/3/4/5
Existing enablers:	Existing programming languages, ontology transformation languages
Required enablers:	Library converters, ontology transformers, ontology governance process

ID - Type	5.15 – Q
Name:	Integrated (information) services
Description:	IOP must support creation of composite (information) services by dynamically merging existing services. Such services can be developed and tested separately from each other, but still operate together as a united whole.
Rationale:	Modularity speeds up development and eases maintenance of applications in a complex environment. Standard services, such as voice recognition or media streaming, are used by many applications.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: high
Relation to Interoperability:	Service interoperability

IOP principles	57 (98)
D5.11-v1.0	Confidentiality Level: PU (Public) 2010-01-09

Related scenarios:	Many, e.g. WP_1.UC_NXP1-Feb-6 WP2.GS1.1-090213-GS1, 2, 3 WP2_UC1_S1,2,3,4 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S1/2/3/4/5
Existing enablers:	Web mash-up techniques, ontology techniques
Required enablers:	Dynamic adaptation to different ontologies

Reliability

ID - Type	5.16 – Q
Name:	Reliable information delivery
Description:	IOP shall be able to deliver reliable information. For example: <ul style="list-style-type: none"> - information about emergency exits, localization, traffic and medical information, safe location maps, and commands to traffic lights and video cameras in emergency scenarios (WP3) - information about possible faults and sensor state in maintenance scenarios (WP2)
Rationale:	The delivered information shall be reliable and available: safety critical information (i.e. medical info, emergency exits, safe locations). Critical information for smart home maintenance (e.g., sensor state, faults). At the same also time real-time info (video streaming) and command and control messages are to be in time.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: high
Space Significance:	Priority for WP3: high
Relation to Interoperability:	Information interoperability (state/control/location) and service interoperability (ad hoc networking, remote medical service, real-time video streaming)
Related scenarios:	WP2.GS1.1-090213-GS1, 2, 3 WP2_UC1_S4 WP2-Smart Lighting.1-16-2-2009-S3 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S4
Existing enablers:	Service discovery, resource mgmt service;
Required enablers:	Event service, dynamic service discovery, situation based performance and reliability mgmt

ID - Type	5.17 – Q
Name:	Detection of context limitations
Description:	There are services whose activation should change context conditions, e.g., adjusting light or temperature according to user-defined preferences. In case there are environmental constraints that prevent such context adjustments, IOP should detect these situations and support alternative strategies/rollback to the previous situation.
Rationale:	Reliability and usability of applications in any given context.
Architectural Significance:	Priority for WP5; low
Application Significance:	Priority for WP6: high
Space Significance:	Priority for WP1-3 high
Relation to Interoperability:	Service and information interoperability
Related scenarios:	Many (indirectly), e.g. WP_1.UC_VTT6-Feb-6 WP2.UC1.0-160209-S_1, 2, 3, 4, 5, 6, 7 WP2.UC2.0-160209-S_1 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S1/2/3/4/5
Existing enablers:	Active DBs Support for context event notification, commit & rollback mechanisms
Required enablers:	Context detection & reasoning mechanisms

ID - Type	5.18 – Q
Name:	Quality of Context
Description:	Context information can be provided by sensors with variable level of accuracy, freshness and granularity (Quality of Context parameters). IOP should provide support for QoC (quantitative) determination from sensors, aggregation and run-time verification.
Rationale:	The verification of QoC levels is needed to ensure that service adaptation is performed based on actual context conditions. In addition, whenever context-triggered events are notified, IOP should provide minimum reliability of triggering information.
Architectural Significance:	Priority for WP5; medium
Application Significance:	Priority for WP6: low
Space Significance:	Priority for WP1-3 low
Relation to Interoperability:	Service and information interoperability

IOP principles		59 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Related scenarios:	many (indirectly), e.g. WP2.GS1.1-090213-GS1, 2, 3 WP2.UC1.0-160209-S_1, 2, 3, 4, 5, 6, 7 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b
Existing enablers:	QoC information provided by sensors (e.g., freshness)
Required enablers:	QoC representation model, support for run-time QoC verification

Extensibility

ID - Type	5.19 – Q
Name:	Loosely coupled extensions
Description:	IOP must support adding extensions to services/applications with minimum programming effort and without changing underlying modules. There must be support for dynamic binding, i.e. adding extensions on a fly.
Rationale:	Cumulative building strategy is often adopted in smart space environments. It makes development more controllable and allows core services to stabilize.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: high
Space Significance:	Priority for WP1-3: medium
Relation to Interoperability:	Service and information interoperability
Related scenarios:	Any scenario where adding services/sensors/personal devices is reasonable, e.g., all scenarios related to WP2.UC1.0-160209; WP2.UC2.0-160209 S_1. WP_3.UC3_S1/2/3/4/5
Existing enablers:	Design patterns, Web mash-up techniques, ontology techniques, extensible programming interfaces
Required enablers:	Ontology extension mechanisms, dynamic adaptation to different ontologies

IOP principles		60 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

2.1.2. Functional Requirements

Polling vs. Event-based

Definition:

This category has been introduced because the possibility of applications requiring “subscribe-notify” support is an important element to consider in the design of the IOP architecture. In fact, it may severely impact on IOP scalability and performance if not handled properly.

ID - Type	5.1 - F
Name:	Polling information
Description:	Application clients shall be able to ask the environment about user's context, e.g., her current location.
Rationale:	Many smart space services need and use context (and in particular location) information for mash-up services, e.g., permitting to change cooler target temperature only in the user's office. Location exploitation as a possible core service in case of Guide-Me application.
Architectural Significance:	Priority for WP5; medium
Application Significance:	Priority for WP6: high
Space Significance:	Priority for verticals: high
Relation to Interoperability:	Information and service interoperability levels
Related scenarios:	WP1_UC_VTT1 WP_2.GS1.1 090213 GS4 WP2_UC1_S1,2,3,4 WP2-Smart Lighting.1-16-2-2009-S1,2,3 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S1/2/3/4/5
Existing enablers:	Location service, location ontology
Required enablers:	Location service, location ontology

ID - Type	5.2 - F
Name:	Context monitoring for event sensing
Description:	The smart space shall be able to monitor context variations to detect if a change happens.
Rationale:	The smart space is able to automatically detect if a device does not work properly or if the set of available/exploited services changes.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: low

IOP principles	61 (98)
D5.11-v1.0	Confidentiality Level: PU (Public) 2010-01-09

Space Significance:	Priority for verticals: high
Relation to Interoperability:	Information and service interoperability levels
Related scenarios:	Many, e.g.: WP1_UC_NOK1 & NOK3 WP2.UC1.0-160209 S_1, S_2, S_3, S_4, S_5 WP2.UC2.0-160209 S_1 WP_2-GS1.1 090213 GS1, GS2, GS3 WP2_UC1_S1,2,3,4 WP2-Smart Lighting.1-16-2-2009-S1,2,3 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b
Existing enablers:	N/A
Required enablers:	Monitoring mechanisms

ID - Type	5.3 - F
Name:	Automatic notification of events
Description:	The smart space shall be able to warn (i.e. notify) users and services if particular events happen
Rationale:	A maintenance operator is automatically informed of a fault. A music player service is automatically informed whenever in the current room there is a hi-fi system it is possible to exploit. Services require to be notified whenever a user changes the exploited application.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: low
Space Significance:	Priority for verticals: high
Relation to Interoperability:	Information interoperability level Environment adapted in relation to the used resources and services
Related scenarios:	Many, e.g., WP1_UC_NOK1 & NOK3 WP2.UC1.0-160209 S_2, S_3, S_4, S_5 WP2.UC2.0-160209 S_1 WP_2-GS1.1 090213 GS1, GS2, GS3, GS4 WP2_UC1_S1,2,3,4 WP2-Smart Lighting.1-16-2-2009-S1,2,3 WP2_CAC1 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b
Existing enablers:	TBD based on SOTA
Required enablers:	Interoperability at the protocol and access control levels (i.e. communication & connectivity)

ID - Type	5.4 – F
Name:	Proactive events
Description:	The smart space shall be able to send proactive events (with advance time with regard to the actual time instant when the interesting condition will be met) when the users of a smart space are in dangerous situation.
Rationale:	< 1 minute for initial detection of emergency; < 45s adding heterogeneous mobile devices to a mobile MESH < 30s adding a mobile MESH device to the MESH
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: high
Space Significance:	Priority for verticals: high
Relation to Interoperability:	Information interoperability level
Related scenarios:	WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b
Existing enablers:	N/A
Required enablers:	Interoperability at the protocol and access control levels (i.e. communication & connectivity)

ID - Type	5.5 - F
Name:	Conflicting requirements harmonization (context reasoning)
Description:	Context reasoning about conflicting requirements in order to harmonize them and achieve an environment which best fits different users.
Rationale:	Different users in the same room may have different requirements, eventually in conflict.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: medium
Space Significance:	Priority for verticals: high
Relation to Interoperability:	Information interoperability levels
Related scenarios:	WP2.UC1.0-160209_S_6, S_7
Existing enablers:	N/A
Required enablers:	Context reasoning and conflict resolution mechanisms as part of information interoperability level

IOP principles		63 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

ID - Type	5.6 – F
Name:	Generic time
Description:	Time stamps of the content of smart space should be aligned and therefore a generic time service is required as a core service of IOP.
Rationale:	Time stamp of content is a property of content that has meaning/constraint/action when it is used for notifying the owner or user of the smart space or when information is mashed up for providing a richer information service.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: n/a
Space Significance:	Priority for verticals: high
Relation to Interoperability:	Information interoperability and service levels
Related scenarios:	WP1_UC_VTT6, VTT7, VTT8 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b
Existing enablers:	SIB, rule-engine
Required enablers:	Time stamp as part of information ontology

Statically known types of data vs. dynamic types of data

Definition:

This category of requirements provides the possibility to integrate also “Dynamic types of data” at runtime (data describing new domain knowledge, e.g., new types of sensors either with new monitoring indicators or with known indicators but encoded in a statically unknown format/syntax).

ID - Type	5.7 – F
Name:	Mashing-up information
Description:	IOP merges different information sources, makes reasoning based on merged information and activates notifications to users and/or devices.
Rationale:	New information can be derived easily by changing the combination dynamically according to the context.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: medium
Space Significance:	Priority for verticals: n/a; high
Relation to Interoperability:	Information interoperability levels
Related scenarios:	WP1_UC_VTT3, ALL2, NXP5 WP2_UC1_S1,2,3 WP2-Smart Lighting.1-16-2-2009-S1,2 WP2_CAC1. WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b
Existing enablers:	Active DBs
Required enablers:	Context reasoning mechanism inside information interoperability level

ID - Type	5.8 – F
Name:	Evolvability of information
Description:	Information production is separated from information consumption
Rationale:	New information sources based on new technologies should be able to establish and at least partial down-ward interpretation should be guaranteed.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: high
Space Significance:	Priority for WP1, WP3: medium
Relation to Interoperability:	Information interoperability level Content (i.e. information) is adapted according to the used resources,

IOP principles		65 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Related scenarios:	WP2.UC1.0-160209 (many indirectly) WP2.UC2.0-160209 S_1 WP_2-GS1.1 090213 (many indirectly) WP_3.UC3_S1/2/3/4/5
Existing enablers:	Ontology transformation languages
Required enablers:	Ontology transformers

Open environment vs. closed environment

Definition:

This category of requirements aims at pointing out the opportunity or not of supporting situations where unforeseen devices (e.g., from unforeseen vendors) needs to connect to the IOP infrastructure in a given smart space.

ID - Type	5.9 – F
Name:	Evolvability of device and service environment
Description:	IOP must present variety of devices (sensors, appliances and actuators) to applications in a uniform way as services, abstracting away the details of the physical world.
Rationale:	Users must be able to remove old and add new devices and services to a smart space environment without the aid of an engineer. Applications should not be dependent on any particular hardware implementation. Developers can define services without having to understand the physical world.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: high
Space Significance:	Priority for WP1-3: high
Relation to Interoperability:	Device and service interoperability
Related scenarios:	Many, e.g. WP_1_UC_VTT6-Feb-6 and WP2.UC1.0-160209 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S1/2/3/4/5
Existing enablers:	Existing devices and adaptors, ontology techniques
Required enablers:	Domain-specific ontologies, adaptation support services

IOP principles		66 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Static deployment vs. dynamic deployment

Definition:

This category of requirements focuses on the opportunity or not of having dynamic deployment of software and metadata descriptions over clients and participating devices. For instance, if a mobile phone needs to get a client application installed on board, should this happen during the set-up phase or dynamically when the SOFIA-based smart space is already running?

ID - Type	5.10 - F
Name:	Dynamic context awareness
Description:	End users shall be able to search information/services/devices according to their preferences from the smart spaces in close proximity and adjust the user's personal space (services, devices) according to the relevant information found, e.g., switching devices and continuing the same application with the same configuration/state seamlessly.
Rationale:	Due to user satisfaction the rules defined by the user itself (i.e. by user preferences) are stronger than the rules of other smart spaces. Adjusting information flow according to user's tolerance. Change of location doesn't impact on user's activity, i.e. the user terminal scans the available (media) devices, selects the one that fits best to user's location and preferences, and continues the service/application in that new device.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: low
Space Significance:	Priority for verticals: high
Relation to Interoperability:	Information interoperability level
Related scenarios:	WP1-UC-NOK1, VTT2 WP1_UC_VTT5 WP1-UC-NOK1, VTT2 WP2.UC1.0-160209 S_1, S_2, S_3, S_4, S_5, S_6, S_7 WP2.UC2.0-160209 S_1 WP2_UC1_S1,2,3 WP2-Smart Lighting.1-16-2-2009-S1,2 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S1/2/3/4/5
Existing enablers:	service discovery
Required enablers:	Dynamic service/information discovery, context reasoning and adaptation mechanisms inside IOP

IOP principles		67 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

ID - Type	5.11 – F
Name:	Automatic synchronization of smart spaces
Description:	Information of smart space shall be always up-to-date, e.g. personal space in auto and mobile phone are synchronized automatically if dynamic synchronization service is activated.
Rationale:	A person can use personal space for different purposes; family, professional, hobby etc. which are handled with different security/privacy level but should be merged into one application, e.g. calendar.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6:n/a
Space Significance:	Priority for verticals: high
Relation to Interoperability:	Information interoperability and service levels
Related scenarios:	WP1_UC_CRF4 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S1/2/3/4/5
Existing enablers:	rule-engines, negotiation protocols
Required enablers:	Mirroring information to many places

IOP principles		68 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

ID - Type	5.12 - F
Name:	Dynamic service discovery
Description:	IOP shall be able to identify the similar services, compare them and notify the user a service that best fits to user's preferences. Alternatively, IOP automatically switches to the more powerful service, eventually even dynamically downloading software components on the client device.
Rationale:	A user is able to use the service at lowest price and best quality; the service may be already active on the infrastructure-side or downloaded and activated on the client-side.
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: high
Space Significance:	Priority for verticals: high
Relation to Interoperability:	Information interoperability and service levels
Related scenarios:	Many, e.g., WP1_UC_CRF5 WP2.UC2.0-160209 S_1 WP_2-GS1.1 090213 GS4 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b WP_3.UC3_S1/2/3/4/5
Existing enablers:	Existing service discovery and context awareness mechanisms
Required enablers:	Dynamic service discovery and dynamic quality mgmt

IOP principles		69 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Legacy

Definition:

This requirement states how the use of legacy devices, systems, software, services should be taken into account.

ID - Type	5.13 - F
Name:	Interoperability between heterogeneous devices/services
Description:	IOP shall be applied to heterogeneous terminals; e. g. wearable PDAs, smart phones, central monitoring stations. IOP shall be able to interact with legacy already available devices/services, integrating them in the SOFIA project
Rationale:	IOP is able to take full advantage of even already available legacy devices, e.g., to monitor and control the behavior of already installed Siemens air conditioners
Architectural Significance:	Priority for WP5; high
Application Significance:	Priority for WP6: medium
Space Significance:	Priority for verticals: high
Relation to Interoperability:	Information and service interoperability levels
Related scenarios:	WP2.UC1.0-160209 S_2, S_3, S_4, S_5, S_6, S_7 WP2.UC2.0-160209 S_1 WP_2-GS1.1 090213 GS1, GS2, GS3 WP2_UC1_S1,2,3 WP2-Smart Lighting.1-16-2-2009-S1,2 WP2_CAC1 WP_3.UC1_S1 WP_3.UC2_S1/2/3a/3b
Existing enablers:	Existing user devices
Required enablers:	Adaptation support services

IOP principles		70 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

2.2. Prioritization of the IOP requirements

This paragraph deals with the criteria to rank the scenarios identified in the different vertical WPs. These criteria of scenarios have been used for prioritizing the IOP requirements.

1) The maximum business impact and 2) the fast and low-risk realization criteria were used to rank individual scenarios. The ranking criteria were specified by the PMT according to the previous analysis work and agreed among the partners.

1. Maximum business impact

- Added value: usefulness to customer is higher than adoption effort and costs
- Partners interest: opens business opportunities for all WP participants
- Market penetration: WP participants can bring the solution to the market

Prioritized scenarios:

- WP1: Enhanced navigation (ALL2) and Mobile user preferences (VTT5)
- WP2: Support for maintenance operators on site (CCC)
- WP3: Outage-triggered operation (maintenance issue in subway trains monitoring (ACCIONA))

2. Fast and low-risk realization

- Availability of technology: compatible with and builds on popular legacy technologies
- Implementation complexity: nature and amount of R&D effort is known and feasible

Prioritized scenarios:

- WP1: Follow me music (NOK), Automatic highest quality music (CRF), Automatic lowest cost Internet connection (CRF), Waking system (VTT), Automatic clock adjustment (VTT)
- WP2: Several scenarios, e.g. Status reminder when leaving, Activation when arriving, Go to sleep/wake up (NOK)
- WP3: Information sharing (NOK)

Additional criteria for scenario prioritization and selection to generate concise and representative classes of scenarios:

3. Support for SOFIA objectives

- Optimal information exchange: amount of required ontologies and quality data is limited
- Ad-hoc composability: run-time, emerging interoperability in use cases, no dependency to interoperability assumed at design-time (tolerance against missing information and services)
- Acceptability: solutions are acceptable by users and society (privacy, security, etc.)
- Common solutions: technology sharing among many use cases, verticals can adapt to or extend it

In the final analysis the following scenarios got the highest priority:

- WP1: Enhanced navigation scenario is the most challenging one, with the highest return on investment.
Mobile use preference has the best market impact, and implementation should be fast.
- WP2: Support of maintenance operators on site and the scenarios that it can exploit: fault management based on automatic detection, activation when user arrives, status check by a passing maintenance person, changing a lamp that is about to die, and super scenario 'semantic connections'.
- WP3: Outage triggered operation (ACCIONA), Dining and buying shoes (NOK) and Subway station (WMC)

Based on this ranking of scenarios, the IOP requirements were prioritized. The following tables summarize the ranking of the IOP quality and functional requirements.

Table 3. High priority quality requirements of IOP

Importance for IOP	Relation to scenarios	Req-ID	Req-Type	Title
HIGH	11 scenarios	5.1-Q	Security	User/device/smart space application authentication
	>10 scenarios	5.2-Q		Access control to smart appliances and authorization
	<10 scenarios	5.3-Q		Shared information integrity and privacy
	>10 scenarios	5.4-Q		Non-repudiability of performed operations & requests
	>10 scenarios	5.5-Q		User/smart space/service immunity
	>10 scenarios	5.6-Q		Security monitoring (auditing)
	8 scenarios	5.7-Q		Intrusion detection (also related to 5.2 – F)
	>10 scenarios	5.8 - Q	Availability	Availability/survivability of IOP, services, and resources
	>10 scenarios	5.9 – Q	Performance	Real-time notification and information delivery (with controlled response time)
	>10 scenarios	5.10 – Q		Scalability of IOP (with respect to users, objects, implementation and information space size)
	>10 scenarios	5.12 – Q	Adaptability	Context-based service adaptation
	>10 scenarios	5.13- Q		Automatic reconfiguration
	>10 scenarios	5.14 – Q	Integrability	Heterogeneous interfaces and implementation languages
	>10 scenarios	5.15 – Q		Integrated (information) services
	>10 scenarios	5.16 – Q	Reliability	Reliable information delivery
	>10 scenarios	5.19 – Q	Extensibility	Loosely coupled extensions

Table 4. High priority functional requirements of IOP

Importance for IOP	Relation to scenarios	Req-ID	Req-Type	Title
HIGH	>10 scenarios	5.2 – F	Event detection and notification	Context monitoring for event sensing
	>10 scenarios	5.3-F		Automatic notification of events
	5 scenarios	5.4 – F		Proactive events
	2 scenarios	5.5 – F		Conflicting requirements harmonization (context reasoning)
	8 scenarios	5.6 – F		Generic time
	>10 scenarios	5.7 – F	Statically known vs. dynamic type of data	Mashing-up information (context reasoning)
	Many indirectly	5.8 – F		Evolvability of information
	>10 scenarios	5.9 – F	Open vs. closed environment	Evolvability of device and service environment
	>10 scenarios	5.10 – F	Static vs. dynamic deployment	Dynamic context awareness
	>10 scenario	5.11 –F		Automatic synchronization of smart spaces
	many	5.12 – F		Dynamic service discovery
	> 10 scenarios	5.13 -F	Legacy	Interoperability between heterogeneous devices/services

As conclusion, the IOP requirements of the highest priority (17 quality requirements and 12 functional/non-quality requirements) are used as architectural drivers which will guide the definition of IOP principles to be followed while architecting and implementing the IOP. Also evaluation criteria will be based on these two sets of requirements. Evaluation criteria will be used as input for evaluating and testing that the implemented IOPs will meet their requirements.

IOP principles		73 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

3. IOP Principles And Criteria To Evaluate The IOP

This section presents the principles of the Interoperability Platform (*IOP*) and the Criteria to evaluate the *IOP*.

3.1. *IOP Principles*

3.1.1. Introduction

SOFIA InterOperability Platform (IOP) is an infrastructure to assist its users with added-value interoperable information about objects existing in the user's environment. It combines *SOFIA* information interoperability solutions with existing/legacy solutions for service and physical level interoperability.

IOP principles originate from the requirements of the vertical WPs scenarios.

IOP speciality is its information interoperability level.

The interoperability is based on common ontology models used in modelling information.

Information may originate from heterogeneous legacy and embedded devices spread in the environment or may be produced by aggregators of *IOP* information.

IOP's information repository may be active and trigger external entities to react to relevant and selected environmental changes.

Such features should be met at high quality level.

Focus is on the following **qualities**:

6. business to be generated
7. its applications engineering process (i.e. design, development and deployment of the applications)
8. the interaction models (interaction among users, their environment and their history)
9. security and dependability (i.e. *IOP* should respect privacy, enforce access policy rules, and assure information integrity and trust)
10. its performance, energy efficiency and scalability

Such an *IOP* is expected to start a market for applications that can interoperate independently from their business/vendor/manufacturer origin, introducing in this way a radical change to the traditional application scenario which is based on fixed business boundaries.

Expected applications may be cross domain, may adapt to the user situation, may spontaneously start when required and have the potential for significant market penetration and socio-economic impact.

IOP principles		74 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

3.1.2. Principles

With the above picture in mind, the following 16 principles are adopted by SOFIA design team and they should be reflected in IOP Big Picture and in IOP implementations.

1. **The shared information principle** (Reqs: 5.1-F, 5.2-F, 5.11-Q).

- The *IOP* manages a shared information search domain called **Smart Space (SS)** accessible and understood by all authorized applications.
- The Smart Space Information is represented in a uniform and use-case independent way (e.g. with RDF triples or with “molecules of triples” or other similar structures).
- Such Smart Space information representation needs to comply with all *IOP* principles
- Information is about the objects existing in the environment or about the environment itself
- Information interoperability and semantics are based on common ontologies that model the information.

2. **The simplicity principle** (Req: 5.8-F)

- The *IOP* deals with information
- The *IOP* information level is use-case agnostic
- The *IOP* consists of a few general and simple components: for example the Smart Space and a secure protocol to access it.

Comments:

- for the *IOP*, information is information – use of information and reasons why information exists are other issues
- Use-case related issues either are modeled with ontologies or are dealt with externally to the *IOP* or both (see principles 3 and 4).

3. **The Service principle** (Reqs: 5.11-F, 5.12-F).

- A **Smart Space** may behave like a service in a SOA, i.e. it may be discovered and accessed as Service offered by a Service Platform.
- The specific service offered is sharing of interoperable Smart Space information.

Each SOFIA application may interface to one or more SSs (through a simple *IOP*-specific interface that may need to be “localized” on the service platform used).

Localization and use case specific functions may be performed at this level, before joining the Smart Space (e.g. credentials based smart space access control).

4. **The Agnostics principle** (Reqs: 5.13-Q, 5.14-Q, 5.19-Q)

- The *IOP* is agnostic with respect to ontology, application programming language, service platform exposing the SS, communication layer and hosting device/system.

5. **The IOP extensibility principle** (5.6-F, 5.19-Q)

- The *IOP* does not provide *a-priori* defined functionalities to manipulate the information, in addition to inserting and removing the information for sharing
- *IOP* functionality may be extended with domain ontologies (stored in the Smart Space) and with information manipulation applications
- if these “applications” become usable through tools of the development environment (i.e. SOFIA IDE) then they will be called “*IOP extensions*” or “*IOP core services*”.

IOP principles		75 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

The following “applications” (mentioned in the TA or proposed by the partners) may lead to the implementation of “*IOP extensions*”:

- T5.3: Security Management Service (Reqs: 5.3-Q, 5.4-Q, 5.6-Q, 5.7-Q)
- T2.5: Energy Management Service (Reqs : 5.8-Q, 5.17-Q)
- T5.2: Context management Service (5.5-F, 5.7-F, 5.10-F, 5.11-F)
- T5.2: Generic Time (Req: 5.6-F)
- T5.2: Real-time information delivery (Req: 5.9-Q) (unless already offered by the “native” SS)
- T5.2: Dynamic service discovery (Req: 5.12-F)
- T5.3 and T5.4: Quality of Service Measurement

6. **The evolvability principle** (5.9-F, 5.5-F).

- The *IOP* should support the addition of new applications as well as applications that are not affected by changes/extensions to the SS scope.

This principle envisages that the *IOP* provides means to implement software modules (e.g. *Knowledge Processors*) that adapt to changes in the Smart Space Knowledge Base, without the need to change their code. For example, if relevant sensor information is added to the SS, the Application should benefit without the need of changing the KPs nor the application. “*IOP extensions*” (see Principle 5)” should meet this principle. As another example, KPs designed to meet Requirement 5.5 - F: Conflicting requirements harmonization (context reasoning), should also meet this principle. It is implied that an Interface between the application and the KP is properly defined.

7. **The context management principle** (Reqs: 5.12-Q, 5.13-Q, 5.18-Q, 5.2-F, 5.10-F) .

- Context management may become an *IOP* extension, according to the “*IOP* extensibility principle”

Smart Space Information is associated to the objects existing in the environment. *IOP* should enable the aggregation of interoperable information into higher level context information, as this is useful in several application scenarios.

As the information returned by the *IOP* depends only from the query (how it is formulated) and from the information that is available, it is left to the ontology level to define context semantics. Context may be managed both at Information and Service level (i.e. internally or externally to the *IOP*).

8. **The notification principle** (Reqs: 5.3-F, 5.4-F)

- Applications may subscribe to be alerted upon a context-change event.

9. **The usability principle** (Req: 5.11-Q)

- User Interaction management may become an *IOP* extension, according to the “*IOP* extensibility principle”

The Usability of SOFIA applications should benefit from the context-dependence principle.

It is left to the ontology level to define the semantics of interaction events. The interaction between the users, their environment and SOFIA applications may be managed both at Information and at Service level (i.e. internally or externally to the *IOP*).

IOP principles		76 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

10. The security and trust principle (Reqs: 5.1-Q...5.7-Q)

- When Smart Space access control is required, it may be handled both at Service Level (i.e. externally to the IOP) and at Information Level"

Appropriate ontologies need to be defined if the *IOP* is required to respect privacy, enforce authentication and access control policies at finer granularity than the Smart Space itself, or if shared information integrity, confidentiality and trust needs to be provided.

Security, privacy and trust management may become an *IOP extension*, according to the *IOP extensibility principle*.

11. The IOP Business model and its implementation principle (Reqs: 5.13-Q, 5.14-Q, 5.15-Q, 5.13-F)

- Development tools and engineering phases of SOFIA applications should be consistent with *IOP* business models and should be mapped to the actors of SOFIA's value chain.

12. The legacy principle (Req: R.13-F)

- Legacy (i.e. existing) devices access and exchange information with the Smart Space through a simple *IOP*-specific, use-case independent protocol. Such exchanged information is modeled by Legacy Domain Ontologies.

Legacy devices may provide information to the Smart Space and may subscribe to the Smart Space to receive information from the SS.

13. The scalability principle (Req: 5.10- Q)

- The *IOP* should scale with respect to the number of users, the number of devices, the resources available on each device, the amount of content of the Smart Space and, the number of SSs.

14. The performance principle (Req: 5.9-Q)

- Performance of *IOP* localizations and of SOFIA applications should be evaluated at development time and should also be measurable at execution time.

The criteria for execution time performance evaluation should be defined by a performance metrics ontology included in the Smart Space.

Performance monitoring applications may become an *IOP extension*, according to the *IOP extensibility principle*

15. The RAS (Reliability, Availability and Serviceability) principle (Req: 5.16-Q)

- Reliability, Availability and Serviceability of *IOP* localizations and of SOFIA applications should be evaluated at development time and should also be measurable at execution time.

The criteria for execution time RAS evaluation should be defined by RAS metrics ontologies included in the Smart Space. RAS management applications may become an *IOP extension*, according to the "*IOP extensibility principle*"

16. The productivity principle

IOP toolkits should support minimum time KP development (Adding this principle to the previous *IOP* principles list was proposed in Eindhoven WP5-WP6 meeting, on Nov 19, 2009). This principle could be met, for example, encouraging KP reusability at SOFIA's *Application Development Kit* level.

IOP principles		77 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Comments to the principles

SOFIA's Big Picture should be checked against the above listed 16 principles.

A Smart Space may be **distributed** for performance, reliability or functional reasons (Principle 1).

SOFIA Information Search domain is expected to be active in the sense that a user may subscribe to be alerted when specified SOFIA information enters the search domain (Principle 6)

Context Dependent Secure Services may be discovered on third party Service Platforms and they may adapt their behavior to the user context and profile, including security/privilege profile (Principles 7 and 10)

Functional Requirement 5.4-F on sending proactive events may not be met by the proposed *IOP* in general terms, but only if the candidates subscribe to receive the proactive events. SOFIA applications (or non-SOFIA services) may send proactive events to "not-subscribers" bypassing the SS. Specifically, with reference to 5.4 F description: "The *smart space* shall be able to send proactive events (with advance time with regard to the actual time instant when the interesting condition will be met) when the users of a smart space are in dangerous situation", it should be noted that with the *IOP* principles stated above an application may receive an event from the SS only after a subscription.

Interoperability of SS-Information is based on the following principles:

1. Information semantics is represented in ontologies expressed, for example, in RDF (Principle 1)
2. An information elementary element is, for example a triple: sub pred obj, where the subject and the object refer to ontologies classes and properties, while the object can refer to an ontology class or it can be a primitive value (Principle 1).
3. A legacy device may generate SS-Information if its native info is captured by a SOFIA entity and translated into SOFIA info (Principle 12)
4. A legacy device may be controlled by the *IOP* through a legacy dependent application which subscribed to or is able to access the smart space (Principle 12).

3.2. Criteria To Evaluate The IOP

3.2.1. Motivation

These evaluation criteria aim to verify if the implemented *IOP* meets the requirements identified by the verticals. It is expected that, if the above stated principles materialized in *IOP* localizations, then the requirements of a very large set of applications, including those related to SOFIA's vertical scenarios should be inherently met.

Therefore - when evaluating an *IOP* – all 16 principles should be checked.

The *IOP* evaluation criteria have been classified into three categories:

1. Functionality related properties
2. Quality properties to be measured at development time
3. Quality properties to be measured at run time.

The main reason for this classification is that the evaluation of these properties has to be done as separate evaluation activities for each category and with different kinds of methods, techniques and, tools.

IOP principles		78 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

3.2.2. Criteria for Functional Evaluation

The first evaluation phase includes checks that IOP principles have been followed and that the functional requirements of high priority have been met. This step may include qualitative and quantitative analysis activities. For example, a qualitative evaluation method is used when the best pattern for achieving the desired IOP principle is to be selected. Quantitative analysis based, for example, on simulations is needed when detection or notification mechanisms are evaluated.

The following features should be covered by the IOP (i.e. these features are mandatory):

1. IOP provides the functionality needed for information interoperability and service interoperability. Information interoperability shall follow the definitions and rules defined by the Big Picture. The service interoperability level provides constructs to handle building blocks of the information interoperability level. (Principles 1-3)
2. IOP is agnostic to used software technologies (e.g. ontologies, programming languages, service technologies, legacy) (Principle 7, R 5.13-F)
3. IOP is extensible; space is extensible, information is extensible, and knowledge interpretations are extensible. Qualitative evaluation method is to be used for extensibility evaluation. (Principle 4, R5.8-F, R5.19-Q)
4. IOP provides a set of detection and notification mechanisms for context sensing, for activating specific functionality and alarming for upcoming events, i.e. reactive and proactive actions activated by changes, data or events should be possible. (R5.2-F...R5.4-F)
5. IOP produces information with relevant indicators of its source and quality of source. (R5.6-F, R5.11-F)
6. IOP provides evolvable information sharing environment, i.e. devices and services can be changed without effect on applications to the extent specified in principle 6 (evolvability principle) (R5.9-F)
7. IOP supports run-time information mash-up. (R5.7-F, R5.10-F, R5.11-F, R5.15-Q)
8. IOP provides mechanism for searching and adapting information relevant for the requestor's purposes if information exists in the smart space and is available for the requestor. (R5.10-F, R5.11-F)

3.2.3. Criteria for Development Time Quality Properties Evaluation

The second evaluation phase concentrates on the quality characteristics that should be covered by designs and implementations and can be evaluated at development time.

- IOP architecture can be easily implemented with different implementation technologies (R5.14-Q)

A. Criteria related to Principle 11 (Security):

Mandatory:

- Identity of users and devices has to be authenticated, when needed. Different authentication levels are supported (e.g. level 0...3) (R5.1-Q)
- Access to smart space is controlled by appropriate countermeasures for users/devices/services. No access without authentication is provided. (R5.2-Q)
- Unauthorized attempts to access to smart spaces are prevented (R5.7-Q)
- Information integrity is proved during transmissions between information sources and sinks. (R5.3-Q)

Optional:

IOP principles		79 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

- Mechanisms for separating personal information from other information is implemented by IOP (R5.3-Q)
- Actions of users and devices are accounted and available for non-repudiability purposes (R5.4-Q)
- Mechanisms for identification and ignoring harmful content (R5.5-Q)
- Security auditing mechanism of IOP supports different security levels (R5.6-Q)

B. Criteria related to Principle 6, 9, 13 and 14 (Performance and Dependability):

Mandatory:

- IOP should keep records on available resources and ensure that smart space is able to continue its operation without losses of resources/failures produced by disasters (R5.8-Q)
- IOP is fully scalable: the number of resources, information providers, and consumers should well scale up to numbers that are compatible with application-specific and deployment scenarios (R5.10-Q)
- Autonomic adaptation of a smart space. Different adaptations are supported; resources, services, information, quality of information/services/resources (R5.12-Q, R5.13-Q)

Optional:

- Real-time notification and information delivery (R5.9-Q)
- Reliable information delivery (R5.16-Q)

3.2.4. Criteria For Execution Time Quality Properties Evaluation

The third evaluation phase addresses the quality characteristics that are observable only during operation. Security, performance and dependability are execution qualities and therefore, the above mentioned classes A (Security) and B (Performance and Dependability) criteria are to be applied for execution time evaluation. However, the metrics to be used in the development time and execution time might be different and different measuring techniques are needed.

The criteria for execution time evaluation shall be defined by a set of ontologies, each of which focuses on one specific quality attribute, such as Security Metrics Ontology, Reliability Metrics Ontology, and Performance Metrics Ontology. These metrics shall be defined in Task 5.3 and used in Task 5.4.

IOP principles		80 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

4. Guidelines to design a Smart Space

This section is the result of the interaction between vertical and horizontal WPs during the first 11 months of the SOFIA project. It is based on the Big Picture terminology and inspired by the experiences on designing Application Specific KPs gained by the partners in 2009. It is intended as a guide for Smart Environment “Governors” that should create and run a Smart Environment.

4.1. Smart Environments deployment strategy

4.1.1. Introduction

It should exist a recognized body responsible for a Smart Environment; such “body” should have rights of ownership for governing and management of the specific “space” where the Smart Environment is deployed.

This body should be responsible for the ontology life cycle consistency and should rely on a set of instruments that specify and document the smart environment. To this end, a number of templates were proposed, that are intended to be verified and refined during SOFIA life.

In Par. 4.2 these templates are motivated and referenced. Some of them are presented in Section 5. It is expected that a comprehensive, optimized, tested, and mature smart environment deployment strategy will be available by the end of the project: this will be one of the main results of the interaction between horizontal and vertical WPs. Therefore, new refined versions of the templates proposed in this section are expected.

Moreover, the initial version of the KP taxonomy is introduced. The KP taxonomy may also evolve when more experience on smart space development has been achieved. The aim of the KP taxonomy is to classify the existing KPs into a set of clusters that help smart space developers to find out suitable KPs for their development work. Thus, the KPs taxonomy also helps in smart space evolution management.

If such a vision is agreed, then T5.1 activities should not terminate with this deliverable, as anticipated in SOFIA TA. T5.1 partners suggest continuing their refinement activity - particularly on KP taxonomy and templates - within task T5.2 on SOFIA Reference Logical Architecture. An updated version of this D5.11 section could be promised for t24, ready to be included, for example as an Appendix to D5.22. This is intended to become an up-to-date input to third year WP6 and to WP7 activities.

4.1.2. Terminology

We here recall relevant terms and concepts from the *Big Picture* [1] of SOFIA Reference Logical Architecture, as these terms need to be used with their exact meaning throughout this section.

From the *Big Picture* we recall, particularly, the concepts of **Smart Environment**, **Information World**, and **Smart Space Application**.

1. A **Smart Environment (SE)** is an entity of physical world that is dynamically scalable and extensible to meet new use cases by applying a shared and evolving understanding of information. The characteristics of SE are:

- at least one *Smart Space* is associated to it
- there are two or more *Knowledge Processors* (typically a producer and a consumer)

IOP principles		81 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Smart Spaces and Knowledge Processors are components of the *Information World*

2. **Information World:** A set of information producers and consumers (Knowledge Processors), a shared information space (Smart Space) and its semantic data model (Ontology)

More specifically:

- A *Smart Space* (SS) is a named search extent of information.
- *Knowledge Processors* (KPs) are information world entities that process information and contribute to (insert/remove) and/or consume (query/subscribe) content according to ontology relevant to its defined functionality. A KP needs one or more partner KPs for useful sharing of content, implying *an agreed semantics*.

We may add at this point that:

- *The agreed semantics ensures Information Level Interoperability and it is represented by an Ontology which needs to be modeled as an RDF graph in order to be accessible for the KPs.*
- *SOFIA proposes its own Core Ontology [1]. Moreover, each vertical WP (i.e., personal space, indoor space, and smart city) has its own domain specific ontology that enhances the concepts and properties of the Core Ontology.*
- *SOFIA Core Ontology might be replaced in the future without affecting the IOP, as - according to the Fourth IOP Principle - the IOP is ontology-agnostic.*

3. **Smart Space Application (SSA):** A subset of KPs in a smart environment that use one or more Smart Spaces as resources (among other resources like displays) to perform a desired functionality. The desired functionality is visible to end users either directly through UIs of KPs or through an enhanced functionality of a legacy application.

We may add at this point that, based on IOP Principle 5 on evolvability:

The functionality of a smart environment usually grows by adding KPs and resources rather than by modifying the existing KPs:

- *A new KP may provide new functionalities*
- *A new KP may add information to the SS; this may have an impact on many SSAs, if their KPs were designed to inherently benefit from SS extensions*
- *Resources added to the SE (e.g. new sensors) may extend the SSA functionality without the need to add new KPs (maybe only adding instances of existing KPs).*

For the above mentioned reasons, for many KPs it may be impossible to tell to which SSA the KP belongs to. And it is also difficult to list the KPs of a specific SSA.

4.1.3. KP development approaches

A crucial property of a SE is its ability to grow in two directions:

1. Extending the SS offer (i.e., the value of the information offered by the SS)
2. Extending the KPs offer (i.e., the range of supported applications).

IOP principles		82 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

It is up to the SE Governing Body to keep this promise; this requires that the following three rules are observed:

- The ontology is kept consistent
- KPs do not violate any of the IOP principles
- From the beginning, KPs are designed to be as reusable as possible (i.e. generic, configurable or, adaptable).

Once designed, a KP may be implemented in two ways, called respectively “*triple-based KP implementation*” and “*triple-blind KP implementation*”, the main difference between the two being the visibility of the RDF graph. As shown in fig 1 left, with “*triple-based implementation*” KPs access the SS directly through language dependent APIs called KPIs (early binding).

With the “*triple-blind implementation*” (fig. 1 right) KPs access the SS through methods offered by ontology dependent convenience libraries generated off-line by an ontology processing toolkit included within SOFIA ADK (late binding to the SS).

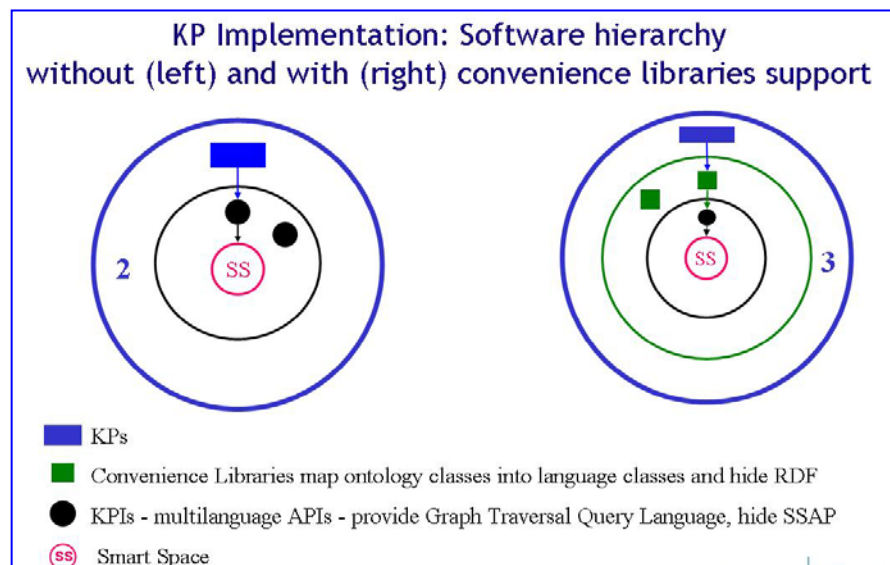


Fig. 1: software hierarchy in KP implementation models

Therefore, KPs may access the SIB through two levels of interfaces: KP Interface libraries (I-level assets) and Convenience libraries (II-level assets).

KP Interfaces (KPIs), as I-level assets, are libraries that access the SS through the “Smart Space Access Protocol” they may be used by:

- KPs that access the Smart Space according to the “triple based approach”
- convenience libraries (II-level assets), which are toolkits that enable KPs to access the Smart Space according to the “triple-blind approach”

Both I-level and II-level interfaces need to be provided by T5.2 for different implementation languages, such as, for example, Java, ANSI-C, C#, Python.

“Triple-blind” development is expected to become the only viable approach while the ontology size increases. But, as the “triple-blind” development requires the availability of SOFIA ADK toolkit, which is currently under development, in the remaining part of this section we shall refer to the “triple-based” approach, which is the only approach extensively used during the first year of the project.

IOP principles		83 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

4.1.4. KP taxonomy

Based on current project progress, and with reference to the IOP principles and to the Big Picture, we may tentatively classify the KPs according to the following “levels”:

1. “*Application specific*” (or “*Domain specific*”) KPs that represent a specific domain (e.g. personal space) or application area (e.g., health monitoring applications).
2. “*Adaptable*” KPs that may be easily and quickly adapted to instantiate new “*domain specific*” KPs (e.g., a WSN manager, a Data Correlator, see WP3’s D3.12)
3. “*Common*” KPs that can be used as such, or may just need some type of parametric reconfiguration, but do not need any code adaptation
4. “*Core*” KPs: these are KPs that may impact the IOP Architecture at the service level, extending the initial version of the IOP Architecture (e.g. the Authenticator and the InterSIB bridge proposed by WP3)

Usually a KP originates as an application-specific KP (it is produced because there is a specific need in a specific scenario), and it has then the opportunity to be promoted through the entire KP category-chain through a generalization process.

The following generalization principle may be proposed:

- I-level: *Adaptable KPs*; an *Application Specific KP* is turned into an *Adaptable KP* when its code is adapted to a new use case
- II-level: *Common KPs*; *Adaptable KPs* originate *Common KPs* when their code is generalized and “parameterized” in such a way that it may be reused in different “scenarios” without any code change. Parameters could be set by a GUI or can be specified by the invoking KP; it is expected that parameters will mostly be ontology nodes. *Common KPs* may become integration elements between domains and they may be used in cross-domain pilots
- III-level: *Core KPs*; only those KPs that are needed in every instantiation of SS will achieve this priority level.

Therefore, once implemented, an *Application Specific KP* becomes implicitly a candidate for a new KP and, as soon as the new KP is available, the originating KP becomes an *Adaptable KP*. In Fig. 2 underneath, Taxonomy of “*Adaptable KPs*” is shown; the leaves of the tree are *Application-Specific KPs*. As soon as a new KP is implemented starting from one of these leaves, it is immediately classified as an “*Adaptable KP*”. As the number of KPs is expected to grow very quickly, the underneath figure may become very large, therefore it will need to be handled on-line as it may shortly become impossible to show it on paper.

While looking into Fig. 2 the following definitions should be applied:

- ADAPTER IS ANY KP THAT INTERACTS WITH THE SS AND WITH SS-EXTERNAL ENTITIES
- AGGREGATOR IS ANY KP THAT INTERACTS WITH ONE SS AND WITH NO OTHER ENTITY
- CONTROLLER IS ANY KP THAT DYNAMICALLY UPDATES SEMANTIC CONNECTIONS IN THE SS
- FILTER IS ANY KP THAT PROCESSES/FILTERS SENSOR DATA BEFORE STORING THEM IN THE SS; IT MAY ALSO BE A “SECOND LEVEL” FILTER, THAT PROCESSES SENSOR DATA ALREADY STORED IN THE SS (E.G. THE CONTEXT FILTER)

Shall KP ontology originate from this analysis? Looking into fig.2, you feel immediately attracted by the idea of building a KP ontology starting from the proposed taxonomy. This could be useful both at development time and at run time, as discussed in next Par. 4.1.5 (Guidelines to further research on KPs).

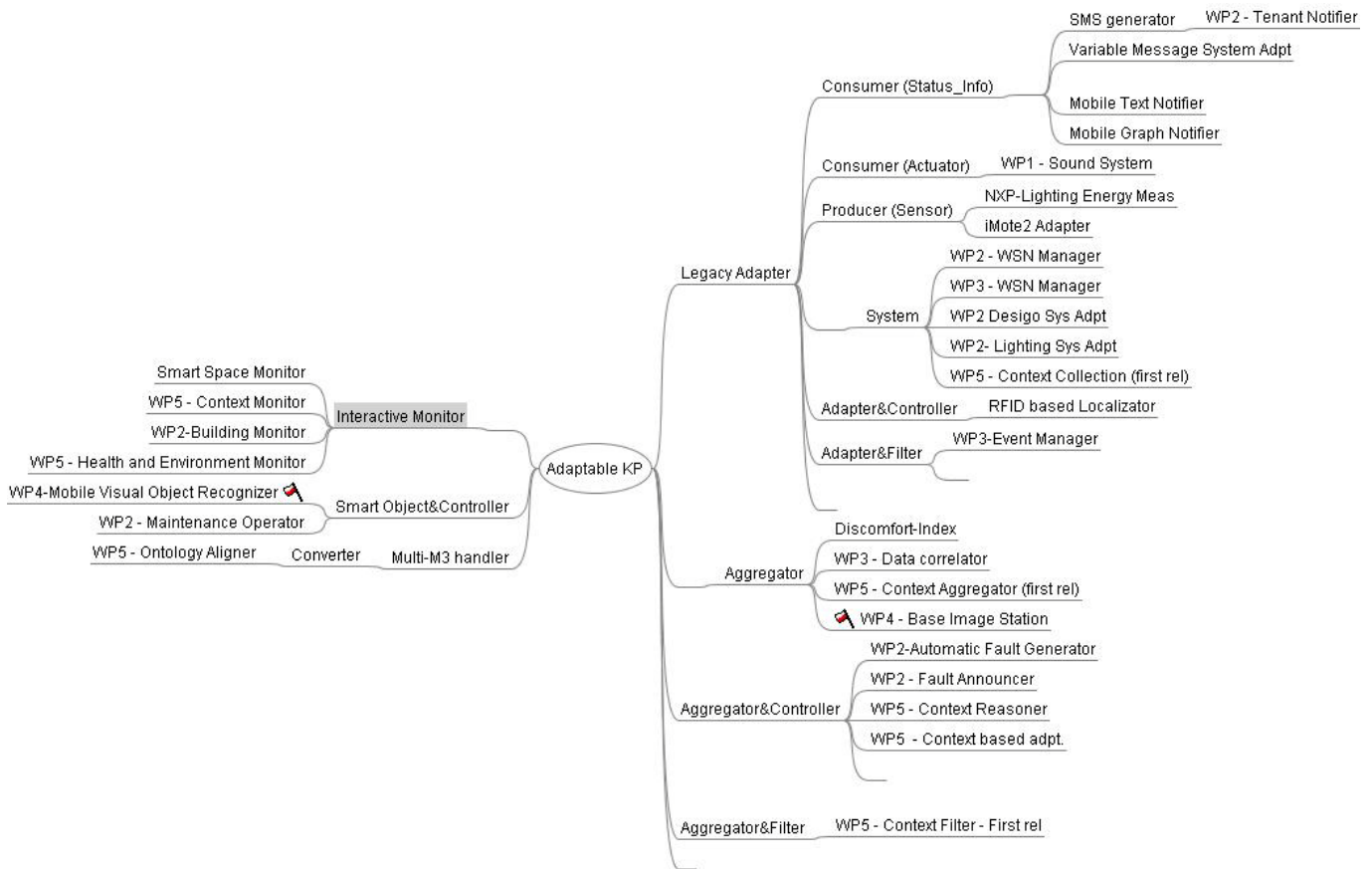


FIG. 2: SOFIA Adaptable KP taxonomy at project time t11; the flags show candidates to become Common KPs

As an example of candidate Common KPs you may consider the flagged KPs in fig. 2: they are components of an Interaction Device consisting of a mobile client that takes pictures, joined to a remote server (Base Image Station) that recognizes the object in the picture. They synchronize through the SS. Such Interaction Device is currently proposed by WP4 as a Building Maintenance Aid, but we may think to parameterize both KPs, in order to use the Interaction device in other domains without changing their code.

Just as examples of candidates Core KPs, we may mention the Authenticator and the InterSIB-Bridge proposed by WP3 (not shown in Fig. 2).

In the future, some context and quality assessment - related adaptable KPs can be refined and introduced as core KPs.

4.1.5. Guidelines to further research on KPs

Based on IOP principles and on the analysis in Par. 4.1.4, further advances on KP knowledge are expected. T5.1 suggests the following list of research questions, which are addressed to KP developers and SS architects.

IOP principles		85 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Question 1: CAN KP ONTOLOGY ORIGINATE FROM THE KP TAXONOMY?

A KP ontology may turn out to be useful at two abstraction levels:

1. At Application Development Kit (ADK) level
2. At IOP Service level

At ADK level, a KP ontology may help developers in finding the best KP to start from (i.e., the existing KP which is the nearest to their needs); it may well happen, for example, that a suitable *Common KP* already exists, so that only a configuration action is required. A KP ontology may help a lot in quickly discovering such a *Common KP*. This would be a further step to meet *IOP productivity principle*.

At IOP service level, a KP ontology may greatly support context-based discovery of KPs. Many localizations of this concept may be envisaged and they should be investigated within T5.2.

For example:

- KP may be turned into services offered by a NoTA RM, and may be discovered through the KP ontology
- KP may be turned into OSGi services and discovered through the KP ontology
- KPs may just be treated as Information in a Smart Space (e.g., the *KP SS*). A “*KP discovery*” KP could be added to the “Core KP” list. Thanks to the KP ontology, the “*KP discovery*” KP would retrieve the required KP URI in the *KP SS*, possibly based on context information.

This should be considered a further step to meet IOP principles nn. 3, 7 (i.e., the Service and the Context-Awareness Principles).

Question 2: IS MUTUALLY EXCLUSIVE ACCESS TO SMART SPACE SUBGRAPHS NEEDED?

In many SS applications a number of users might take advantage from a coordination action automatically performed by a number of associated KPs. For this to occur, the KPs have to perform mutually exclusive transactions each consisting of a sequence of appropriate subgraph searches and updates. Many examples of this way of conduct are expected to be proposed by the vertical WPs and an interesting use case (automatic coordination among many maintenance operators) was already proposed by WP2.

To support these usage models, an atomic Query&Update primitive is proposed and it should behave as the “test&set” instruction in traditional Instruction Set Architectures. Thanks to such a primitive, a “semaphore” data type could be defined and implemented in the IOP to protect the access to “critical subgraphs”, a “critical subgraph” being a subgraph that needs mutually exclusive access. Starting from this example, the IOP synchronization needs should be analyzed in terms of parameters and semantics before to agree on an implementation plan with T5.2.

The proposed synchronization support should not jeopardize the IOP ability to meet the IOP principles. Particularly, principles 2 and 4 (i.e., the “Simplicity” and the “Agnostics” principles should hold; therefore a solution where a class “semaphore” is introduced in the Core Ontology might not be acceptable.

The proposed synchronization support would be a further step to meet IOP principles 9 and 7 (i.e. the Usability and Context Awareness principles).

IOP principles		86 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

Question 3: IS SS DISTRIBUTION REQUIRED?

It may happen that a KP requires info from more than one SS, or it may happen that multiple SS need to merge into one SS, either statically or dynamically. This type of requirement is currently investigated by the verticals WPs. They need to be considered by T5.2, once the requirements are clearly stated.

These future results need to comply with all IOP principles, and particularly, with IOP principles 2 and 13 (i.e., the Simplicity and the Performance Principles). They would be considered further steps to meet IOP principles 1 and 14 (i.e., the Information Sharing and the Scalability Principles).

Question 4: HOW COULD WE FURTHER ENHANCE THE KP DEVELOPMENT PRODUCTIVITY?

Par. 4.1.4 provides guidelines towards the definition of a service ontology and also suggests the reuse of “*Adaptable KPs*” whenever a new KP needs to be developed. But other means to speed up KP development may be devised. For example a set of reusable code patterns may be collected and offered by SOFIA ADK. This already happens - even if informally - when KPs are developed according to the *triple-based implementation*” approach (par. 4.1.3). It has already been verified, in fact, that SS subgraphs are often navigated and searched through very similar pieces of code. Collecting, describing and, properly disseminating these patterns could be a further step to meet the already recalled IOP principle 16 (i.e., the Productivity Principle).

The above questions suggest some of the topics that should be investigated, both at the Architecture and Tool Levels (i.e., by T5.2 and WP6) to further meet the IOP Principles and enhance the IOP qualities.

4.1.6. Smart Environment design steps

Starting from the KP taxonomy proposed in Par 4.1.4, the following SE design steps are envisaged:

1. The smart environment and the initially addressed scenario are defined as follows:
 - The smart environment is described as suggested in Par 4.2.1
 - The initial scenario may be described using the scenario template [6], as already done by all vertical WPs [2], [3], [4]
2. The Domain Ontology is created as an enhancement of the Core Ontology provided by WP6
3. An initial KP list and a KP vs. ontology table is created (see example in tab. 1)
4. The KP coordination is defined
4. KPs are classified based on the proposed taxonomy (Par. 4.1.4) or selected from the KP repository with help of the KP taxonomy
5. All KPs are defined according to the KP template shown in Par. 5.2 (some of the partners have already defined their KPs using a similar template [5])
6. The KPs are developed concurrently based on the previous steps
7. The KPs are connected to the Smart Space by using the KPIs. The KPIs are selected based on the hosting platform and are specified in the KP template

If the IOP principles are met, then - according to the evolvability principle - at a later time:

- new producers can be added to the benefit of the existing KPs
- new consumers may be added that benefit from the existing and future smart space configuration

KPs need to be consistent with the IOP. The criteria to verify whether a KP is consistent or not consistent with the IOP, are expected to be defined by T5.4.

TAB. 1: AN EXAMPLE OF THE *KP VERSUS ONTOLOGY* TABLE FOR A POSSIBLE SMART SPACE

Smart Environment		WP2 Smart Indoor Spaces – Maintenance scenario in large buildings					
Entity subgraphs →	Environment	Data	Fault	Corrective Intervention	Maintenance Operator	Tenant	Maintenance Parameters
KP names ↓							
<i>Automatic Fault Generator</i>	1 Queries	3 Queries and Subscribes to	4 Creates				2 Queries
<i>WSN Manager</i>	1 Queries	2 Creates / Updates					
<i>Fault Announcer</i>			1 Subscribes to	2 Creates 3 Subscribes to 4 Removes			
<i>Building Monitor</i>	2 Queries	2 Queries	2 Queries	1 Subscribes to 3 subscribes to	2 Queries		
<i>Maintenance Operator KP</i>				2 Subscribes to 3 Insert 4 Insert 5 Insert	1 Queries		
<i>Tenant Notifier</i>				Subscribes to			

4.2. Describing a Smart Environment

A smart environment should collect information from the environment and store it into a Smart Space for unanticipated multi-domain and cross-domain *Smart Space Applications*, which are - according to the Big Picture – any digital entity providing known functionality for the user.

Smart Space Applications (SSAs) consist of a subset of KPs in a smart environment and they access one or more Smart Spaces as resources to perform a desired functionality. The desired functionality is visible to end users either directly through UIs of KPs or through an enhanced functionality of a legacy application (which is an application that is not able to directly access a Smart Space).

In order to completely characterize a smart environment, a minimum number of templates and questionnaires should be devised. The following is a possible list:

- ⇒ A “Smart Environment Template”, i.e., the template to describe the smart environment and its extent (see Par. 4.2.1)
- ⇒ A “scenario template” [6] i.e. a template to describe a usecase and the associated scenarios, from which the SSAs may be derived (see Par. 4.1.2, subpar 3)
- ⇒ An “IOP requirements template” (Par. 5.1)
- ⇒ A “KP Template” (Par. 4.2.2 and Par. 5.2)
- ⇒ An “Interaction Device Template” (WP4, [7])
- ⇒ A “User Experience Feedback Questionnaire” (WP4, [8])

4.2.1. Hints for a “Smart Environment Template”

A “*Smart Environment Template*” should be devised to gather the following information:

1. What is the Smart Space for?
2. Who are the expected users?
3. What business model is applied?
 - a. Ownership
 - b. Roles of actors in Smart Space
 - i. The KP company
 - ii. The Service Provider
 - iii. The Information Producers
 - iv. The Application Providers
 - c. Business plans of different actors
4. A reference to the Ontology current version (if available)
5. Who is governing the ontology
6. What information is collected?
7. From which environment:
 - ⇒ physical (e.g., a building)
 - ⇒ logical (e.g., a community of users or a social community)
 - ⇒ virtual (e.g., a virtual model)
8. Which repositories are needed
9. Just a few examples of possible applications
10. Just a few examples of Interaction Devices
11. Just a few examples of context information that could be gathered by dedicated KPs

IOP principles		89 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

4.2.2. Motivation and structure of the “KP Template”

The template herewith envisaged addresses *Domain Specific and Adaptable KPs*.

It is partitioned into 4 Tables, as shown in Par. 5.2.

The KP template purposes are:

- Providing a KP overview, at the level of “executive summary” (Table 1)
- Providing a detailed information-level specification (Table 2)
- Providing a guideline to its implementation and deployment (Sequence Diagram, Table 3)
- Relating KP properties to IOP principles (Table 4)

Table 1 will be collected by T5.1: they are intended to be used for defining the KP Taxonomy (Par. 4.1.4).

Table 4 is used in T5.4 for assessment of the developed KPs and assuring that IOP implementations follow the principles and criteria defined for IOP.

Tables 2 and 3 are only intended as a guide to KP implementation and performance-level description.

The proposed template could be adapted to describe *Common KPs*.

We have not analyzed yet the issue of defining a proper “*Core KP*” template. As it is expected that there will be only few *Core KPs*, defining a new template for a few set of KPs does not bring much added value for smart space environment development.

4.3. References

1. D6.11 Core Ontology, http://www.sofia-project.eu/system/files/SOFIA_D6.11-CoreOntology-v10-2009-06-26.doc
2. D1.11 Personal Space Scenarios and Ontology, http://www.sofia-project.eu/system/files/SOFIA_D1.11_Personal_Space_Scenarios_And_Ontology_v15-2009-04-28.pdf
3. D2.11 Use case and related resources/services: Requirements and specifications, http://www.sofia-project.eu/system/files/SOFIA_D2-11-Use+cases-final-2009-03-31.pdf
4. D3.11 Use case and related resources/services: Requirements and specifications, http://www.sofia-project.eu/system/files/SOFIA_WP3_D3.11_SmartCity_Use_Cases_2009-04-28.zip
5. D3.12 - Smart City Architecture, http://www.sofia-project.eu/system/files/SOFIA_D3-12_Smart_City_Architecture_v1.4.doc
6. Requirements Specification Process - Step 1 - Capturing Use Cases And Scenarios From Vertical Wps http://www.sofia-project.eu/system/files/T5+1_template_for_collecting_use_cases_and_scenarios_090204.doc
7. D4.31 - Preliminary Design for Interaction Data Storage, Enrichment and Retrieval
8. User Feedback Questionnaire, http://www.sofia-project.eu/system/files/SOFIA_User_Feedback_Questionnaire-Final_0.doc

5. TEMPLATES

5.1. IOP Requirements Template

This template specifies the functional and quality requirements to be used as driving factors in the conceptualization and development of the IOP. IOP requirements were extracted from the Vertical Scenarios during the first six months of SOFIA project (see requirement list in Section 2).

Requirement ID - Type	Type: Q for Quality, F for Functional (e.g. : 5.2 – Q)
Requirement Name:	Access control to smart appliances and related authorization
Description:	<p>Short textual description - e.g.:</p> <p><i>When a user/service tries to access any device/smart appliance (e.g., light adjustment in a smart home), IOP should check that it is allowed to access that appliance, based on its (administrative or network) domain and/or other relevant conditions, e.g. location or other contextual conditions.</i></p> <p><i>This requirement also regards any software update installation, which should not breach access control.</i></p> <p><i>Enforcing access control might require support for authorization.</i></p>
Rationale:	<p>Motivation for the requirement - e.g.:</p> <p><i>Access control is needed to prevent unauthorized usage of smart space appliances by services that do not have proper credentials to do so.</i></p> <p><i>This is important to ensure the correct behavior of smart home appliances and to keep automated appliances under control.</i></p>
Architectural Significance:	<p>Motivated impact on IOP architecture design - e.g.:</p> <p><i>Priority for WP5: high (access control solutions should be considered during architectural design)</i></p>
Application Significance:	<p>Motivated impact on IOP Application Development Kit design – e.g.:</p> <p><i>Priority for WP6: medium (applications can rely on the existence of access control support)</i></p>
Space Significance:	<p>Relevance for verticals – e.g.:</p> <p><i>Priority for WP2, WP3: high</i></p>
Relation to Interoperability:	<p>Interoperability levels affected - e.g. :</p> <p><i>Information and service level interoperability</i></p>
Related scenarios:	<p>List of use cases and scenarios exhibiting this requirement – e.g.:</p> <p><i>Samples from personal spaces scenarios:</i></p> <p><i>WP_1.UC_NOK1-Feb-6</i></p> <p><i>WP_1.UC_VTT5-Feb-6</i></p> <p><i>Samples from smart indoor spaces scenarios:</i></p> <p><i>WP2.UC1.0-160209-S_1, 2, 3, 4, 5, 6, 7</i></p> <p><i>WP2_UC1_S1,2,3,4</i></p> <p><i>WP2-Smart Lighting.1-16-2-2009-S1,2,3</i></p>
Existing enablers:	<p>Resources and concepts that could be used to support this requirement e.g. :</p> <p><i>Authorization lists, / group (e.g. only for family members) / role based (e.g. only for caretakers) authorization</i></p>
Required enablers:	<p>Resource needed and concepts that need to be implemented to satisfy this requirement – e.g.:</p> <p><i>Support for collecting context information, access control policy-based system</i></p>

IOP principles		91 (98)
D5.11-v1.0	Confidentiality Level: PU (Public)	2010-01-09

5.2. Template proposal for Application Specific and Adaptable KPs

The template purposes are:

- to provide a **KP overview** (Table 1- Executive Summary)
- to provide a **guideline** to its implementation and deployment (Table 2)
- to provide a detailed **specification** of KP-SS interaction (Table 3)
- to relate KP properties with the IOP principles (Table 4)

Table1 is the reference source to classify the KP within T5.1 KP taxonomy (see Par. 4.1.4).

It is also intended as an additional step of vertical scenarios description, on top of the scenario template, and it tells which requirements the IOP need to satisfy in order to support the KP.

Table 2 adds design/implementation info on top of the existing scenario documentation (i.e., to the scenario template). It contains information useful to the SS governor and should encourage new applications that may use the interoperable data produced by the KP. It is intended to be filled by the implementation team. It also include an optional section (still to be refined) aiming to evaluate KP impact on Smart Space scalability and performance.

Table 3 is intended to detail the interaction between the KP and the associated Smart Spaces. It describes the Information Level behavior of the KP, and it is needed as the interaction between KP and Smart Space has an impact on SS consistency. Table 3 has not the structure of a traditional table, but it is a sequence diagram, which is a much more expressive description style for interaction sequences.

Table 4 relates KP properties with the IOP principles and is intended as a support to evaluate KP qualities. It is also intended as a support to evaluate if the KP may be a candidate “Common” or “Core” KP. This template may be easily modified to describe also “Common” KPs: it should be enough to add a section to describe its parameters.

In order to show how the template should be used, an example of an already filled template is now presented. The selected KP is a Smart Space Translator that collects specific information from one Smart Space (upon notification) and copies such information in a second Smart Space after aligning to the destination Smart Space Ontology. BTW this KP is part of the Demo that won the First Artemis Exhibition Award (Artemis Autumn Event, Madrid, October 29-30 2009).

Table 1: Executive Summary (Service to be provided and IOP requirements)

Overview	
Name	<i>SmartSpace_TRANSLATOR (From "VTT SS" TO "UNBO SS")</i>
Purpose	<i>Collecting specific sensor data from VTT SIB and copying them into UNBO SIB. Sensor data are: temperature and humidity in a living room and a cellar</i>
Proposer	<i>T5.2, UNBO and VTT, Federico Spadini</i>
Justification (<i>Where and when needed and why</i>)	<i>Designed for Sofia demo at Artemis Autumn Event in October 2009</i>
Legacy adapter	<i>No, it is not a legacy adapter</i>
Addressed scenario	<i>Demo scenario: may health parameters be better understood if related to environmental conditions?</i>
SIB Interaction	
Type	<i>KP is Consumer for VTT SIB and Producer for UNBO SIB</i>
Interaction paradigm	<i>Callback/Event-based: the KP subscribes to VTT SIB for new specific sensor data</i>
IOP Properties needed	
Functional (<i>List of the required IOP functional properties see Par. 2.1.2</i>)	<i>5.3 – F - Automatic notification of events 5.6 – F Generic time: needed for time-stamping data, non for scheduling polling actions 5.7 – F - mashing-up of information – this KP provides UNBO SIB with information that may be used by mashing-up services 5.8-F- Evolvability of information: this KP aligns VTT SIB sensor data to UNBO SIB sensor data through ontology translation: it surrogates 5-8–F in a specific case 5-11 F - Automatic synchronization of smart spaces This KP surrogates in a particular case 5-11 – F, which is not provided yet by the IOP</i>
Quality (<i>List of the required IOP quality properties see Par. 2.1.1</i>)	<i>5.8 – Q - Availability/survivability of IOP, services, and resources – Service should be available despite possible failures in single sensors 5.9 – Q - Real-time notification and information delivery (with controlled response time)- controlled response time not yet available 5-10 – Q Scalability - with respect to number of locations and number of sensors</i>
Candidate as "adaptable" KP	
<i>"Adaptable" KPs may be used to easily and quickly instantiate application specific KPs</i>	
YES	
If YES , describe how and in which other scenarios the KP can be used	<i>This KP may inspire any KP that needs to align a "master" SIB to a "slave" SIB, whenever the involved SIBs do not share their ontology</i>
Is the KP based on other "adaptable" KPs? [Yes/no]. If yes list the "adaptable" KPs used (e.g. a WSN manager, a Data Correlator, see D3.12)	<i>NO</i>

Table 2: KP Implementation and Deployment Guide

Version Number	V. 1.0
Date of current release of this KP	25/10/2009
Validation level	alpha
Information Level and Semantic Interoperability	
Used ontology (<i>Clear identification</i>)	UNBO SIB: UNBO Sensor Ontology -file_name: UNBO_ARCES_sensor_ontology_sep_09.owl VTT SIB: VTT Sensor Ontology
Ontology Awareness Required: <i>Main concepts used (e.g. main classes and properties)</i>	ARCES Sensor_Platform and its properties ARCES Environment ARCES Sensor_Data and its properties: ⇒ Unit of measure ⇒ Value ⇒ TimeStamp ⇒ Environment Contains Entity Vtt Sensor, and its properties ⇒ hasLocation ⇒ hasUnitType ⇒ hasValue ⇒ hasName Data And its properties
Expected reasoning support from the IOP	Automatic spanning of subclasses
Information published on the Smart Space	Published on UNBO SS: ⇒ Initially the instances of the desired VTT SS sensors are created in the UNBO SS ("ontology update") ⇒ Then the UNBO SS is updated with the sensor_data subscribed to the VTT SS
Information consumed from the Smart Space	Consumed from VTT SS: ⇒ when both smart spaces are joined successfully, a function is launched which queries the VTT smart space for all available sensors and properties ⇒ then a subscribe to the desired VTT sensor data is made and upon notification of new data, a function is called from the subscribe (to maintain atomicity and synchronization) and the UNBO smart space is updated
Information exchanged externally to the Smart Space (legacy information or services required and/or provided to the external world)	
Required from outside the SS at run time	none
Provided outside the SS at run time	Diagnostics messages to the shell console
KP Configuration data (<i>e.g. initial data for the KP</i>)	Smart Space Name, IP address and Port for both SSs (these are initial "KP configuration data")
System Level and Portability	
Target hardware platform	Any Linux machine having access to both SSs
Run-time environment	Mac OS/X separate TCP connectivity to two SSs in two separate subnets using Multi-Homing
Programming language	Python

KP API	<i>Python TCP KPi</i>
Development environment	<i>TextMate</i>
Run time GUI required? If yes, why? If yes, describe its purpose, e.g. for initial configuration, runtime interaction, input values, output values. Also tell which graphics libraries are used, e.g. QT	<i>QT based Interface to provide initial "KP configuration data"</i>
Service Level	
<i>How the KP is launched and how the KP joins its SIB</i>	
KP activation	<i>KP is activated manually</i>
SIB discovery: Does the KP automatically discover its SIB(s) on a service platform? If yes, which one? e.g. NoTA RM, UDDI or OSGi.... If no, which information is needed to join its SSs?	<i>KP is not able to discover its SIBs; the user must provide name, IP address and port of joined SIBs (these are "KP configuration data")</i>
SIB access framework	<i>TCP</i>
Core KP needed? If yes please specify which ones	<i>No</i>

Local data (not shared): data needed by the KP and not shared with the SIB	<ul style="list-style-type: none"> ⇒ <i>a dictionary of sensor URIs based on VTT Smart Space is created.</i> ⇒ <i>This dictionary has all information from the VTT smart space regarding the sensors (such as URI, type of sensor, value, unit of measure)</i> ⇒ <i>this information is then parsed to match locations for "cellar" and "living room" and to match units of measure; in this way sensor of interest in the VTT SS are identified</i>
---	---

The following Tab. 2 entries are optional and need further refinement from the partners

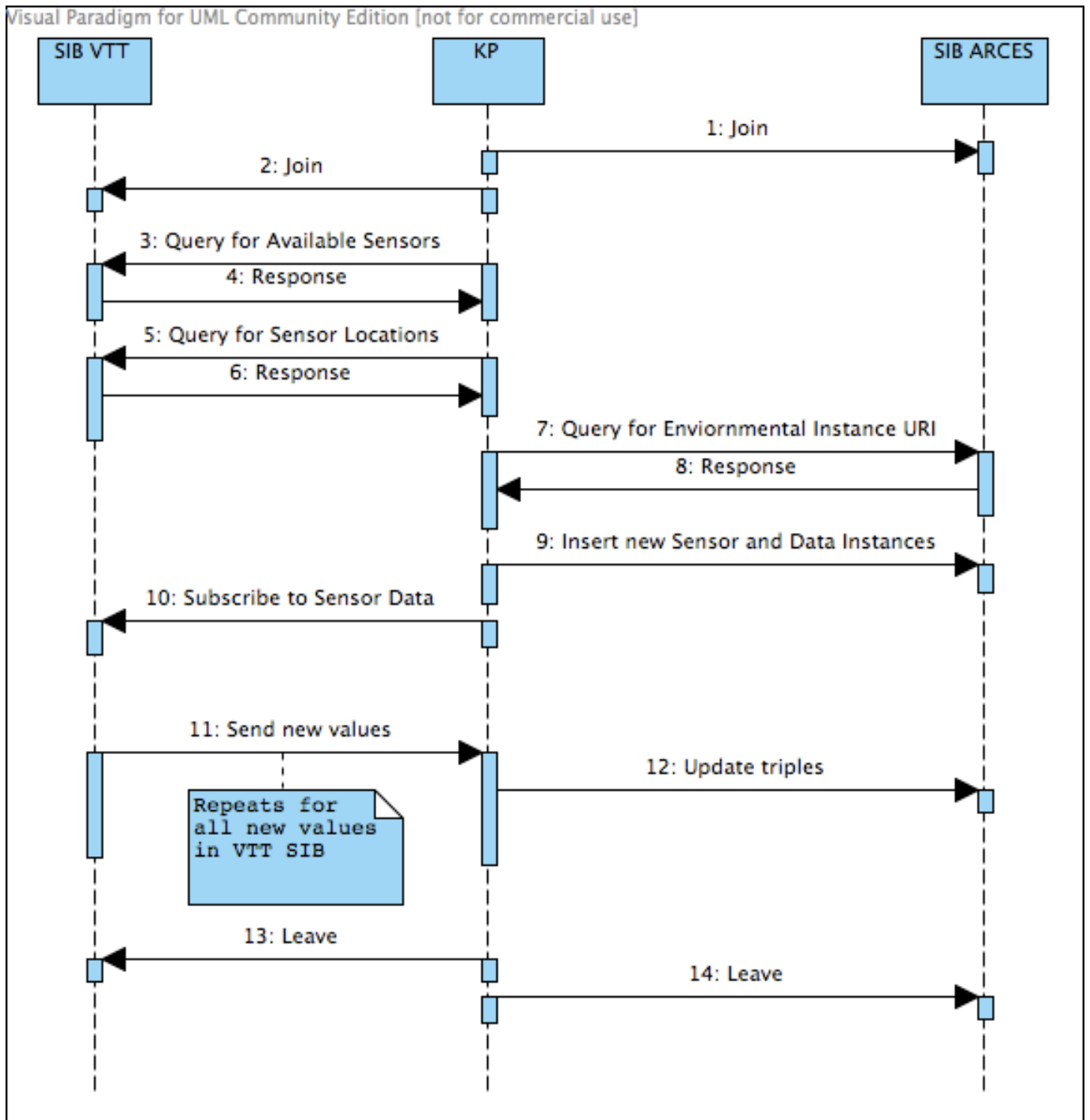
Methods to understand KP impact on SS scalability and performance	
Amount of Transactions/Time Unit	<i>e.g. 10 transactions per minute This is an indicator of the impact on both communication and computational load</i>
Number of SIB accesses per transaction <i>Partners are invited to suggest a better indicator</i>	<i>e.g. amount of queries or triples retrieved from the IOP per transaction</i>
Scaling factors of most critical transactions <i>Partners are invited to suggest a better indicator</i>	<i>e.g. main nodes to be retrieved scale linearly with respect to the number of instances of a specified entity (number of sensors in a room, number of people in the environment,...)</i>
Number of subscribes	<i>e.g. number of subscribes equal to number of customers</i>
Any other indicator, please describe	<i>e.g. amount of triples inserted in the IOP</i>

Tab. 2 purpose: This table adds design/implementation info on top of the existing scenario documentation (i.e., to the scenario template).

Table Source: it is intended to be filled by the implementation team.

Table Destination: It contains information useful to the SS governor and should encourage new applications that may use the interoperable data produced by the KP.

Table 3. Sequence diagram to specify KP interaction with the SS



This sequence diagram details the Interaction between the KP and the associated Smart Spaces

Table 4: Analysis of the KP properties against the principles/criteria

Principles/Criteria		Enforcing level <i>e.g., may not be achieved in current IOP version, may be achieved only when based on OSGi, will be achieved in the future,</i>
Information interoperability	<p><i>Is this KP needed for achieving information interoperability between devices? Principles 1...2.</i></p> <p><i>Does this KP need a modification to the ontology specified in tab. 2?</i></p>	<p><i>Not between devices but between SIBs: This KP aligns SIBs that are not interoperable because they use two different ontologies. It meets two goals: Interconnecting two SIBS</i></p> <p><i>Aligning their Ontologies</i></p> <p><i>Neither of the above functionalities is currently provided by the IOP</i></p>
Service interoperability	<p><i>Does this KP provide/use services for achieving service interoperability between devices? Principle 3.</i></p>	<p><i>NO, only data</i></p>
Technology agnostic	<p><i>Up to which extent is the KP technology agnostic?</i></p> <p><i>Principle 4</i></p>	<p><i>It requires Python on Linux, and the Python-TCP-KPI</i></p> <p><i>It assumes two specific ontologies</i></p>
Extensibility	<p><i>Does the KP support possible extensions/legacy to services/applications with minimum programming effort and without changing underlying/existing modules/services?</i></p> <p><i>Principles 5 and 12.</i></p>	<p><i>It may be easily modified to adapt to other ontologies</i></p>
Evolvability and context-awareness	<p><i>Does the KP adapt its behavior according to user's or SS situation or to some specific event notified?</i></p> <p><i>Principles 6 and 7.</i></p>	<p><i>NO</i></p>

Usability	<p><i>Is the KP be self-explanatory, non-intrusive, easy-to-learn and easy-to-use for the end-user?</i></p> <p><i>Principles 5 and 9.</i></p>	<p><i>e.g., what is it needed to configure this KP to a specific situation?</i></p>
Security and Trust	<p><i>Does the KP produce/consume information with high level of accuracy/security?</i></p> <p><i>Principle 10.</i></p>	<p><i>e.g., security requirements were not considered and are left to the architecture (service and information levels)</i></p>
Performance	<p><i>Is performance of KP known and is it scalable?</i></p> <p><i>Principles 13 and 14.</i></p>	<p><i>e.g., see tab. 3</i></p>
Productivity	<p><i>Is this KP to be developed from scratch or may existing KPs be reused?</i></p> <p><i>Might this KP be reused in other Smart Environments?</i></p> <p><i>Principle 16</i></p>	
In summary: Are all of the relevant IOP principles met by this KP?	<p><i>e.g., YES</i></p> <p><i>e.g., NO, but it does not jeopardize the IOP architecture</i></p>	
Is this KP a candidate "Common" or "Core" KP?	<p><i>For the definition of "Common" and "Core" KPs, please see Par. 4.1.4 on KP Taxonomy</i></p>	

This table provides input to the KP evaluation process.

APPENDIX: IOP requirements quick reference guide

The underneath table lists all quality and functional requirements described in Section 2.

IOP Quality Requirements	
ID-Type	Name
Security	
5.1 – Q	<i>User/device/smart space application authentication</i>
5.2 – Q	<i>Access control to smart appliances and related authorization</i>
5.3 – Q	<i>Shared information integrity and privacy</i>
5.4 – Q	<i>Non-repudiability of performed operations and requests</i>
5.5 – Q	<i>User/smart space/service immunity</i>
5.6 – Q	<i>Security monitoring (auditing)</i>
5.7 – Q	<i>Intrusion detection (also related to 5.2 – F)</i>
Availability	
5.8 – Q	<i>Availability/survivability of IOP, services, and resources</i>
Performance	
5.9 – Q	<i>Real-time notification and information delivery (with controlled response time)</i>
5.10 – Q	<i>Scalability of IOP</i>
Usability	
5.11 – Q	<i>Usability of end-user services (uniform interface and presentation of semantic connections)</i>
Adaptability	
5.12 – Q	<i>Context-based service adaptation</i>
5.13 – Q	<i>Automatic reconfiguration</i>
Integrability	
5.14 – Q	<i>Heterogeneous interfaces and implementation languages</i>
5.15 – Q	<i>Integrated (information) services</i>
Reliability	
5.16 – Q	<i>Reliable information delivery</i>
5.17 – Q	<i>Detection of context limitations</i>
5.18 – Q	<i>Quality of Context</i>
Extensibility	
5.19 – Q	<i>Loosely coupled extensions</i>
IOP Functional Requirements	
Polling vs. Event-based	
5.1 – F	<i>Polling information</i>
5.2 – F	<i>Context monitoring for event sensing</i>
5.3 – F	<i>Automatic notification of events</i>
5.4 – F	<i>Proactive events</i>
5.5 – F	<i>Conflicting requirements harmonization (context reasoning)</i>
5.6 – F	<i>Generic time</i>
Statically known types of data vs. dynamic types of data	
5.7 – F	<i>Mashing-up information</i>
5.8 – F	<i>Evolvability of information</i>
Open environment vs. closed environment	
5.9 – F	<i>Evolvability of device and service environment</i>
Static deployment vs. dynamic deployment	
5.10 – F	<i>Dynamic context awareness</i>
5.11 – F	<i>Automatic synchronization of smart spaces</i>
5.12 – F	<i>Dynamic service discovery</i>
Legacy	
5.13 – F	<i>Interoperability between heterogeneous devices/services</i>