WoO - Web of Objects Project

D3.3 Reference Architecture

••••••••••••••••••••••••••••••••••••••••••••••••••••••

|  |  |  |  |
| --- | --- | --- | --- |
| Contract Number: | | ITEA-2 10028 | |
| Starting date: | 01/01/2012 | Ending date: | 31/12/2014 |

|  |  |
| --- | --- |
| Title of the main deliverable: | Reference Architecture |
| Title of the main deliverable: | WoO-WP3-D33 |
| Work Package deliverable: | D3.3 |
| Task related to the deliverable: | T3.3,T3.4,T3.5 |
| Type (Internal / Restricted / Public): | Internal |

|  |  |
| --- | --- |
| LEADER | CONTRIBUTORS: |
| **Lead Participant:** Thales Communications and Security  **Lead author:** Nicolas Dumont,   Mario Lopez Ramos  **E-mail:** [nicolas.dumont@thalesgroup.com](mailto:nicolas.dumont@thalesgroup.com),   [mario.lopezramos@thalesgroup.com](mailto:mario.lopezramos@thalesgroup.com) | Youssouf Mhoma (Thales)  Pierre Châtel (Thales)  Vicente Hernández (UPM)  Maria Cinta Saornil Gómez (Visual Tools)  Guy Myoung Lee (Institut Telecom)  Ismail Salhi (UMLV)  Juan Vera del Campo (UPC)  Marc Caballe May (Telespazio)  David Excoffier (Sogeti)  Khaled Elsayed (Cairo University)  Ahmed Khattab (Cairo University)  Jean-François Goudou (TS)  Arnaud Galimberti (TS)  Daniel Gaston (Prodevelop)  Christophe Joubert (Prodevelop) |
| REVIEWERS |
| Mihaela Brut (TS)  Patrick Gatellier (TS) |

HISTORY

NB: a **status** is associated to each step of the document lifecycle:

* **Draft**: this version is under development by one or several partner(s);
* **Under review:** this version has been sent for review;
* **Issued**: this version of the document has been submitted to EC.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Status | Date | Author | Main Changes |
| 0.10 | Draft | 22/06/2012 | Nicolas Dumont | ToC proposed at Madrid meeting |
| 0.11 | Draft | 12/12/2012 | Nicolas Dumont | ToC updated |
| 0.12 | Draft | 21/01/2013 | Nicolas Dumont | ToC updated and attribution of sections |
| 0.13 | Draft | 04/03/2013 | Nicolas Dumont | ToC updated  Contributions from TCS and Spanish demonstrator section |
| 0.14 | Draft | 27/03/2013 | Nicolas Dumont | Contributions from Visual Tools, Institut Télécom, Technical University of Madrid (UPM), University Paris-Est Marne-la-Vallée, Thales Communications & Security, Telespazio, UPC, Prodevelop |
| 0.15 | Draft | 23/04/2013 | Nicolas Dumont | Contribution from Visual Tools, Institut Télécom, UPM |
| 0.16 | Draft | 21/05/2013 | Nicolas Dumont | Integration of contributions about   * French demonstrator architecture (Sogeti) * Wo0 system lifecycle (UPM) * Egyptian demonstrator (Cairo University)   Style fixes |
| 0.17 | Draft | 12/06/2013 | Nicolas Dumont | Integration of contributions about:   * “Application creation frontend”, in overall architecture (UMLV) * Service Choreography and Orchestration (UMLV) * Generic Gateway design (TCS) |
| 0.98 | Pre-Review | 02/07/2013 | Nicolas Dumont | Update French demonstrator description (TCS & TS) |
| 0.99 | Pre-  Review | 30/07/2013 | César  González | Update Components. |
| 1.0 | Issued | 15/11/2013 | Jean-François Goudou | Update document and final review |

TABLE OF CONTENTS

[1 References 5](#_Toc372304770)

[1.1 List of acronyms 5](#_Toc372304771)

[1.2 Glossary of Terms 5](#_Toc372304772)

[1.3 List of Tables 5](#_Toc372304773)

[1.4 List of Figures 5](#_Toc372304774)

[2 Introduction 7](#_Toc372304775)

[2.1 Public Summary 7](#_Toc372304776)

[2.2 Purpose 7](#_Toc372304777)

[2.3 Mapping to other tasks 7](#_Toc372304778)

[2.4 Analysis of architectural drivers 8](#_Toc372304779)

[3 Overall architecture 11](#_Toc372304780)

[3.1 Functional view 12](#_Toc372304781)

[3.2 Layers and functional building blocks 13](#_Toc372304782)

[3.2.1. Device 13](#_Toc372304788)

[3.2.2. Communication 13](#_Toc372304789)

[3.2.3. Management 14](#_Toc372304790)

[3.2.4. Service 16](#_Toc372304791)

[3.2.5. Security 17](#_Toc372304792)

[3.2.6. Application 17](#_Toc372304793)

[3.3 Logic and functional introduction 17](#_Toc372304794)

[3.3.1. Objects & Services registry 17](#_Toc372304796)

[3.3.2. Service discovery and selection 18](#_Toc372304797)

[3.3.3. Service exposure 18](#_Toc372304798)

[3.3.4. Service execution platform 20](#_Toc372304799)

[3.4 Models 22](#_Toc372304800)

[3.4.1. Description of Objects and their capabilities 23](#_Toc372304802)

[3.4.2. Semantic Description of Objects and their capabilities 25](#_Toc372304803)

[3.4.1 Description of Sensing Devices 27](#_Toc372304804)

[3.4.2 Semantic Profile of a Service 29](#_Toc372304805)

[3.5 Ontology description 30](#_Toc372304806)

[3.5.1. Device/Actuator 30](#_Toc372304808)

[3.5.2. Resources 31](#_Toc372304809)

[3.6 System lifecycles 32](#_Toc372304810)

[3.6.1. Service 32](#_Toc372304812)

[3.6.2. Device 33](#_Toc372304813)

[3.7 Security 35](#_Toc372304814)

[3.7.1. Security requirements 35](#_Toc372304816)

[3.7.2. Security components 37](#_Toc372304817)

[3.7.3. Distributed networks 39](#_Toc372304818)

[3.8 Components 42](#_Toc372304819)

[3.8.1. IPv6 Infrastructures 42](#_Toc372304821)

[3.8.2. Service Framework for the Web of Objects 44](#_Toc372304822)

[3.8.3. Resources Management 47](#_Toc372304823)

[3.8.4. Distributed service infrastructure 50](#_Toc372304824)

[3.8.5. Smart streaming protocol 53](#_Toc372304825)

[4 Domain specific architectures refinement 56](#_Toc372304826)

[4.1 Open Smart Neighborhood Demonstrator 56](#_Toc372304827)

[4.2 Egyptian Demonstrator 57](#_Toc372304828)

[4.2.1. Overall System Architecture 57](#_Toc372304832)

[4.2.2. Information Model 58](#_Toc372304833)

[4.2.3. Functional Architecture 59](#_Toc372304834)

[4.2.4. Deployment Architecture in Detail 62](#_Toc372304835)

[4.3 French Demonstrator 66](#_Toc372304836)

[4.3.1. Information Model 71](#_Toc372304838)

[4.3.2. Functional Architecture 71](#_Toc372304839)

[5 Conclusion 73](#_Toc372304842)

# References

## List of acronyms

## Glossary of Terms

|  |  |
| --- | --- |
| WoO | Web of Objects project |
| Architectural Drivers | These consist of functional requirements, quality attributes, business constraints and technical constraints that can potentially 'shape' the architecture. |
| WSN | Wireless Sensor Network. |

## List of Tables

[Table 1 Sorted architectural drivers table 9](#_Toc368582543)

[Table 2 Communication layer functions 13](#_Toc368582544)

[Table 3 Management layer functions 15](#_Toc368582545)

[Table 4 Service layer functions 16](#_Toc368582546)

[Table 5 Security requirements 30](#_Toc368582547)

[Table 6 Mapping security component to requirements 32](#_Toc368582548)

[Table 9 : Sensor Nodes Functional Architecture 56](#_Toc368582549)

[Table 10 : Actuator Node Functional Architecture 57](#_Toc368582550)

[Table 11: Gateway Node Functional Architecture 58](#_Toc368582551)

[Table 12 : Server Functional Architecture 59](#_Toc368582552)

[Table 13 : End-user Device Functional Architecture 59](#_Toc368582553)

## List of Figures

[Figure 1 Relationship between the tasks of WP2 and WP3 7](#_Toc360550712)

[Figure 2 Overall architecture 10](#_Toc360550713)

[Figure 3 Service exposure architecture 19](#_Toc360550714)

[Figure 4 Service discovery - First Phase Use Case 20](#_Toc360550715)

[Figure 5 Service discovery - Second Phase Use Case – Connector generation 20](#_Toc360550716)

[Figure 6 Service composition via Orchestration (left) or Choreography (right) 21](#_Toc360550717)

[Figure 7 Ontology overview 22](#_Toc360550718)

[Figure 8 Device/Actuator class inside the ontology. 23](#_Toc360550719)

[Figure 9 Resources class inside the ontology 23](#_Toc360550720)

[Figure 10 Activity diagram for execution lifecycle. 25](#_Toc360550721)

[Figure 11 Sequence diagram for manual registration 26](#_Toc360550722)

[Figure 12 Sequence diagram for automatic registration 26](#_Toc360550723)

[Figure 13 Sequence diagram for composed services registration 27](#_Toc360550724)

[Figure 14 Sequence diagram for manual un-registration 27](#_Toc360550725)

[Figure 15 Sequence Diagram for automatic un-registration 28](#_Toc360550726)

[Figure 16 Semantic service registry, discovery, selection - Use Case Diagram 35](#_Toc360550727)

[Figure 17 Service discovery and selection - Components diagram 36](#_Toc360550728)

[Figure 18 Gateway functional overview 40](#_Toc360550729)

[Figure 19 Conceptual design 41](#_Toc360550730)

[Figure 20 UPM Use Case diagram 42](#_Toc360550731)

[Figure 21 UPM Components diagram 42](#_Toc360550732)

[Figure 23 Egyptian Demonstration Deployment Architecture 45](#_Toc360550733)

[Figure 24 Service Architecture 46](#_Toc360550734)

# Introduction

## Public Summary

This deliverable describes the first version of Web-of Objects architecture. It defines the main functions offered by WoO system, and introduces proposed models.

## Purpose

The purpose of this document is to describe WoO architecture and preliminary architectures of demonstrators. The architecture is technology agnostic and is dedicated to:

* Overall Web of Objects vision
* Definitions of functions that will be mandatory or optional in Web of Objects
* Remind models proposed to describe entities of WoO (see D3.2 Domain model)
* Proposition of components carrying those functions
* WoO System lifecycles
* Security

A section will describe components that will implement WoO architecture. This part contains only a short description of the components, and is actually a bridge between functions of the architecture and concrete implementation in WP4 & WP5.

Finally, last section is dedicated to domain specific demonstrator’s architecture, with more technical details, and selected technologies.

This deliverable is the first iteration, and will be updated later in the project based on feedback of implementations and experimentations.

## Mapping to other tasks

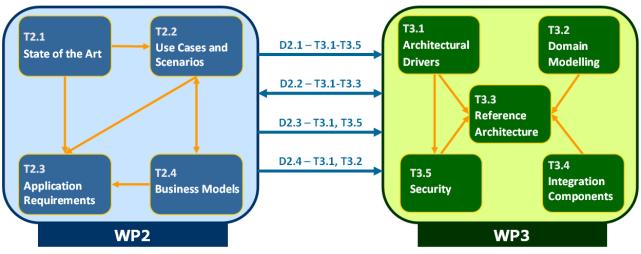


Figure 1 Relationship between the tasks of WP2 and WP3

The results of WP2 have served as input for the different tasks of WP3 as shown in Figure 1. On the one hand, the state of the art developed in task T2.1 provided information about the innovative available technologies that can be used to build the architecture of the Web of Objects (T3.3), including the review and analysis of different existing architectures, requirements and evolution of wireless sensor networks, security mechanisms and modelling tools.

On the other hand, the use cases defined in task T2.2 drive the development of the domain model and the design of the reference architecture of tasks T3.2 and T3.3. Besides, the resulting domain model and the common architecture provide feedback to refine the use cases of task T2.2.

Furthermore, a set of common requirements and technical constraints were extracted from the requirements of each particular use case (T2.2, T2.3) that were used to define the architectural drivers for the architecture, including security requirements (T3.1, T3.5).

Finally, the business models developed in task T2.4 provided business constraints and context for the architecture and the domain model of tasks T3.1 and T3.

## Analysis of architectural drivers

Task 3.1 analyzed requirements and use cases to establish a list of quality attributes requirements (cf D3.1). For each of these, a stakeholder priority mark and complexity mark has been determined. In the following table, those metrics were combined with the formula above to determine priority in Web of Object architecture design. Low means high priority for stakeholders and low complexity. So this table will help to determine which topics are mandatory and which are secondary.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Quality Attribute | Description |  |
| QAS15 | Security (Authentication) | The system must be able to authenticate users to prevent anonymous and malicious access | 1 |
| QAS16 | Security (Authorization) | The system must be able to identify users based on the access level they are granted and provide them relevant information accordingly | 1 |
| QAS19 | Usability  Maintainability | The system works as documented | 1 |
| QAS02 | Scalability | The system must be capable of managing thousands of objects at a time | 2 |
| QAS04 | Manageability | Devices must be able share data and semantic | 2 |
| QAS06 | Security | The system must provide secure communication mechanisms | 2 |
| QAS09 | Robustness | Ability to overcome communication failures | 2 |
| QAS11 | Manegeability | Physical Identification of core system devices | 2 |
| QAS13 | Manageability | User must be able to send current location | 2 |
| QAS14 | Manageability Extensibility Security | The user profile must be secure, accessible and updatable | 2 |
| QAS20 | Extensibility | The format of the data exchanged between the system and the different nodes/devices must be platform-independent | 2 |
| QAS01 | Extensibility | The system must be prepared to add new autoidentification mechanisms/technologies | 3 |
| QAS07 | Manageability Extensibility | Autoidentification of objects during deploy | 3 |
| QAS03 | Manageability | The system must be able to detect/interoperate with heterogeneous and legacy systems | 4 |
| QAS08 | Availability | Ability to detect and react to black-out events | 4 |
| QAS10 | Robustness | Tracing of important system events (Auditing) | 4 |
| QAS18 | Scalability | Since scalability makes reference to adding different functionalities to a system in an escalating fashion, it is only natural to deemed here as the possibility of handling the increasing amount of workload as nodes, services and other resources are added up –or depleted- in a Wireless Sensor and Actuator Network. Obviously, scalability refers not only to the mere addition of new elements, but also to the correct performance of them all. Therefore, it is a non-functional property that will be considered closely to other ones, as robustness, availability or extensibility, for they have to be born in mind if a system is desired to be scalable enough. A middleware architecture based on agents can be useful here, as it will adapt new software elements in an almost seamless manner. | 4 |
| QAS05 | Interoperability Performance | The system must be prepared to exchange a large amount of information | 6 |
| QAS17 | Robustness | Robustness is the ability of an element to keep on working regardless of how hostile its current conditions may be. In the context of the Web of Objects project, robustness is closely linked to Wireless Sensor and Actuator Networks. Here, a subsystem will be regarded as robust if it is able to fare under high traffic volume, low battery level or failing node conditions in a satisfactory way. In order to obtain an acceptable degree of robustness, several mechanisms can be implemented; for example, a self-healing procedure capable of having active motes assuming roles belonging to flawed ones would provide some welcomed resilience, and a preemptive functionality, where nodes will be aware of the precise limitations of other members of the WSAN and will assume whatever task if needed would be useful as well. | 6 |
| QAS12 | Scalability Manageability | System must be able to scale up/down at runtime | 9 |

Table 1 Sorted architectural drivers table

D3.1 proposes also recommendations on technical constraints and business constraints. Technical constrains recommend the use of standards for the description of devices, especially for video related devices. This constrains does not impact reference architecture design, because we it has to be domain agnostic, but it will be nevertheless useful for implementation choices of use cases implying video elements. D3.1 proposes also the adoption of Service Oriented Architecture as design constrains. This constrains has to be leverage by proposed reference architecture. In addition, adopting a full REST architecture is advised.

# Overall architecture

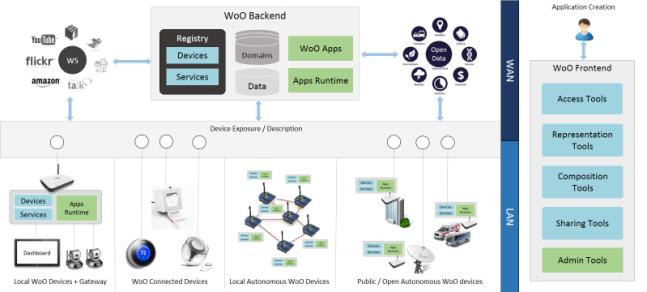


Figure 2 Overall architecture

Web of Object proposes an overall architecture enabling the integration of any connected object into WoO solution. In this section we introduce WoO overall vision, which is not the architecture, but helps to understand the scope of work in WoO. We distinguish two main levels:

1. Local: it concerns the connections of objects at LAN level. Challenges of this level are:

* object connection,
* capabilities discovery,
* logic execution,
* exposure of features in shared formalisms and technologies

1. Backend: this level concerns infrastructure backend that supply WAN connectivity. WoO backend will provide features enabling the design, execution and support of IoT based application. Key aspects will be

* discovery of objects at WAN level
* understanding of and access to functional and non-functional capabilities of connected objects
* logical execution (centralized or decentralized)
* technical and semantic adaptation of objects and exchanged data for cross-domain interaction
* interaction of WoO with external services (Social Networks, Public Open Data, public or private Web Services)

1. Frontend: this architectural block allows users and administrators to control, orchestrate and create applications using service composition and management tools. Key challenges here are :

* Nodes initialization, configuration and maintenance
* Providing node representation tools that could facilitate the application creation process
* Designing composition tools that allows automated service orchestrations and choreographies

An implementation of WoO architecture does not have to fulfill all of those characteristics. We could imagine isolated networks of objects, working together, but only at local level, with or without gateways, if application domain requires such isolation. In this case, WoO brings enablers for smart interactions, based for example on choreography.

We distinguish different characteristics for objects that will determine how we can integrate them into WoO solution:

* **Connectivity**: some objects will be able to connect and be visible directly at a WAN scale (e.g. IPv6 connected devices, …). Some other will be only accessible to LAN levels and consequently will need a gateway (software or hardware) to communicate over WAN (e.g. IEEE 802.15.4 devices, …) Some other will not have any communication means, and will need a connected sensor to be perceived and represented in digital world (e.g. table, box, …).
* **Energy constraints**: some devices are not connected to energy network, and that implies constrains for functions implementations. A tradeoff between performance and battery life duration has to be made. This characteristic has a strong impact on connectivity: for long battery life, objects will have to use specific protocols, and probably rely on a gateway for some functions.
* **Behavior configuration**: Some objects will offer mechanisms to reconfigure their behavior, or even to host applications. With such mechanism, WoO applications could be distributed over a group of objects working autonomously together, organized as choreography. Other behaviors could be accomplished as orchestration, with a conductor responsible of application logic execution.
* **Self-Descriptive**: To enable discovery, capabilities (functional and non-functional) of objects will have to be announced to WoO solution. Such announcement could be done autonomously by objects, through specific protocols, or eventually by embedding this mechanism in a gateway though which connectivity is realized. These capabilities should be described technically to enable technical use of these objects, but also semantically, to enable recommendations or even automatic peering of objects. Those two levels of description are also a solution to heterogeneity: a technical adaptor can be generated to enable communication between two objects having a semantic interest.

## Functional view

Taken into account the ongoing works concerning IoT, the following figure 1 depicts a functional overview of the WoO reference architecture, explained in detail in further sections in this document.



Figure 1.- Functional view.

## Layers and functional building blocks

According to the six initially identified layers, and considering internal building blocks in each of them, this section will provide a brief explanation for each layer, just as a general perspective for the main objective.



### Device

This layer involves all kinds of devices in charge of gathering or metering information from the environment, communicating with others devices, doing actuation on the environment, etc. Each of them will provide different means and capabilities for interacting or communicating with other WoO architecture artefacts.

### Communication

This layer considers every issue regarding the communication between devices and the rest of layers except security and management. The latters could need to directly access device resources.

Brief functional blocks description

|  |  |  |
| --- | --- | --- |
| ID | Name | Description |
| 1 | Gateway | What is called heterogeneous networks can be linked by numerous Local area networks (LAN) and wide area networks (WAN) through gateways. Therefore, gateway bridge the communication among different networks and enforce protocol translations and address translations needed to cross network borders. Gateway have desperate interfaces that can link to different Local Area Network.  For example, bridging between wireless sensor networks with traditional communication networks or Internet, a gateway plays an important role in the applications, which facilitates the seamless integration of wireless sensor networks and mobile communication networks or Internet, as well as the management and control with wireless sensor networks. |
| 2 | Addressing | Identify the devices; make any devices be reachable in order to enable devices to be discovered in the system. This means that through the use of a set of technologies, all the objects are able to interact and cooperate with each other. This can be achieved by efficient routing protocols that provide the functions of data transmission, energy efficiency and so on. In addition, assigning addresses for the devices is taken into account so that all the devices can be identified by the unique address, such as taking consideration of IPV6 technology. Moreover new protocols and technologies should be used to ensure that low power devices can be addressed so that they can expose their services. |
| 3 | QoS | It refers to quality of communication services. Low throughput, dropped packet, errors, jitter, and latency should be taken into account in the communication layer realization.  Quality of service is the ability to provide different priority to different applications, users, or data [flows](http://en.wikipedia.org/wiki/Flow_(computer_networking)), or to guarantee a certain level of performance to a data flow. For example, a required [bit rate](http://en.wikipedia.org/wiki/Bit_rate), [delay](http://en.wikipedia.org/wiki/Network_delay), [jitter](http://en.wikipedia.org/wiki/Packet_delay_variation), packet dropping probability and/or bit error rate may be guaranteed. Quality of service guarantees are especially important if the network capacity is insufficient, especially for real-time [streaming multimedia](http://en.wikipedia.org/wiki/Streaming_multimedia) applications. A network or protocol that supports QoS may agree on a [traffic contract](http://en.wikipedia.org/wiki/Traffic_contract) with the application software and reserve capacity in the network nodes. |

Table 2 Communication layer functions

### Management

This layer combines all functionalities that are needed to govern the system.

Brief functional blocks description

|  |  |  |
| --- | --- | --- |
| ID | Name | Description |
| 1 | Configuration | Contains the system configuration, the components configuration, the device configuration and so on. For example, when the system encounters faults or errors which affect the normal operation of the system, the configuration management should enable to configure or reconfigure the system in order to maintain the normal operation of the system. Providing that the faults or errors make the whole network topology changed, all the devices affected should enable auto-configuration to find a way aiming to maintain the system performance. In addition, the configuration among each component plays a part that all the components in the system can collaborate together to serve the whole system. |
| 2 | Energy Optimization | It aims to manage energy consumption in the system and it could be implemented from the low layer to the higher layer containing service layer, communication layer and management layer. In fact, energy optimization mainly focusses on the communication layer in consideration of the fact that most of the energy cost is due to this layer; and optimal routing protocols can play an important role in saving energy. Besides, energy efficiency should also be considered in services layer and management layer itself, such as how to discover services, how to compose services, how to execute services, how to configure the system, how to manage fault etc., taking consideration of energy optimization. |
| 3 | Publish/Subscribe | To reduce unnecessary data transmission, event-driven mechanism can be used in data collection and transmission in the network. Event-driven mechanism provides capabilities for publish (announces of event generation) and subscribe (announces for event consumption). In order to coordinate the event consumer and the event providers, event servers should exist. Inside of event server, there are event management, event dispatching, and subscription management and subscription checker. The event management is responsible for event storage, event aggregation; the subscription management is in charge of storing subscription information; the subscription checker checks the content of events against the subscriptions; once the events match the subscription, the event dispatching takes charge of sending the content of event to the corresponding destinations. |
| 4 | Fault | It is used to identify, locate, correct and record different kinds of faults that maybe occur in the system. The fault module is responsible to check the fault in network and repair the faults to improve the capability of toleration failure to build a self-diagnostic, self-healing, self-protecting and self-sustaining network. |
| 5 | Context | The context module aims at efficiently capturing multiple context knowledge from/to different layers. The interpretation of the sensed information greatly depends on the context and for efficient processing and communication; context awareness can play a major role. Based on the interpretation of the actual status information from the physical environment and user information, the whole management system might be optimized. The various physical and digital information sources, the users’ profiles and situational preferences, provide an effective context processing environment so that the system is capable of offering more valuable services. Context information can also be used in communication layer to enable routing & addressing devices with high energy efficiency. |

Table 3 Management layer functions

### Service

An open homogeneous distributed service infrastructure is introduced according to the functions and characteristics of the overall architecture. In addition, a context-aware service adaptation layer is formed targeting all the smart objects that can collaborate together to accomplish assigned tasks. Within the service infrastructure, service & device registry, service discovery & look up, a semantic and adaptive service composition and service execution platform are introduced. This section is just a general description of functional blocks, since section 3.3 (logic and functional introduction) will represent that in detail.

Brief functional blocks description

|  |  |  |
| --- | --- | --- |
| ID | Name | Description |
| 1 | Service & Device Registry | Service & Device Registry provides the functions that describe and publish the services and devices to potential consumers (clients). A Service & Device Registry allows you to register information about services and devices with a semantic way that provides facilities to publish and discover services and devices. Database can be used to store the registered information about services and devices. And the mode of registry is divided into static registry and dynamic registry. |
| 2 | Service discovery & Look-up | Service discovery & Look-up allows discovery of services offered by devices and further to access to the services. It means that all the services registered in the system can be discovered and selected automatically according to both the meaning of requests and the semantics of services. |
| 3 | Service Composition | Service composition enables to construct complex and composite services from atomic services to achieve a speciﬁc task. Service composition models can be categorized into centralized and distributed ones. In other words, service composition can be classified in to service orchestration and services chorography. Services orchestration means that each player in the orchestra strictly follows instructions from the central conductor.  Services chorography means different bricks of logic are deployed on each node to perform a global task following a global scenario without a single point of control. |
| 4 | Web of Object Services | Web of Object Services, on the one hand, it provides different kinds of platforms which enable the services to be executed, such as application programming interfaces (APIs) following RSET and OSGI platform based on Proxy server. On the other hand, it provides a platform (Agent) where autonomous services run according to the role of node it takes in the system. |

Table 4 Service layer functions

### Security

This layer defines the security components for the system that provide a safe and reliable way to access the system, and ensure its security and its privacy. All the components can be divided into two parts. One part is related to service security which contains authorization; identity management, trust as well as authentication. The other one is about communication security in which key exchange and management is taken into consideration. It also defines security mechanisms that help protect the network from possible attacks that may occur.

### Application

Application known as an application or an app is computer software designed to help the user to perform specific tasks. Depending on the activity for which it was designed, an application can manipulate text, numbers, graphics, or a combination of these elements.

We believe that a unique solution composed of a limited set of technologies is not feasible in todays and future IoT landscape: evolution is so quick and applications domains are so wide that an adaptive solution is required to tackle heterogeneity. That is why we propose this reference architecture, mainly influenced by state-of-the-art IoT architecture, as a catalogue of main functions that will be accomplish by adapted technologies, and distributed to different components according to application domains and objects characteristics.

## Logic and functional introduction



### Objects & Services registry

Service & Device registry are with respect to both objects registry and services registry.

WoO will provide a semantic registry feature, directly linked to service discovery and selection, based on semantic meta-data **(QAS04)** attached to resources, devices , “Objects” and so on **(QAS03, QAS22)**. A “Service” will either be *manually published in the registry, or automatically registered in the registry* (through automatic discovery feature).. For static registry means that using static mode to register services and devices, and that can be done before the system runs. Another way is to register services and devices dynamically, that is to say, when the system is running, once a new device is detected by the system, the device and its corresponding services can be dynamically registered. And in contrast, if a device does not work anymore, the device and its related services should be automatically canceled.

Besides, the Service & Device registry should provide another function. For example: If a client wants to access a temperature sensor ‘123’ and requests to the Services & Device registry. The Services & Device registry provides all the temperature services to the client, but does not contain the temperature sensor ‘123’. At that time, the Services & Device registry should enable to provide the address of the service which provides by other platform.

In both cases functional syntactic information (service core provided features) and semantic meta-data (providing voluntarily additional – computable – information on the service) is stored at the semantic registry level for further exploitation **(QAS04)**. In addition, database will be used to store the information of Objects & Services registry, and the data structure is provided by T3.2.

### Service discovery and selection

Based on the aforementioned semantic service registry, WoO will be able to manage automatic and semi-automatic (i.e. user driven) heterogeneous and legacy service discovery and selection **(QAS03)**. As shown previously, after publication of a service in the registry, a “Client” can be *automatically registered in the registry* as a consumer of a particular service feature (available or not) and/or *search for a particular capacity endpoint*.

The additional abstract concepts related to the architecture of the semantic *service discovery and selection* are:

* **Search capacity endpoint**: A client has the possibility to search with meta-data information capacities and to get the endpoint of the service proxy (connector) implementing these capacities.
* **Invoke connector**: A client has the possibility to invoke a specific connector deployed by the discovery component
* **Automatic client registration in network**: if a client complies with a discovery protocol supported by the server, it can detect automatically the discovery server and registers to it.

### Service exposure

Providing and consuming information should be there in order to develop a coordinated automated environment. Functionalities available in the devices need to be exposed so that other users and objects can use them. It is called service exposure. Service oriented approach seems to be a promising solution to interconnect objects each other. Each object can expose its functionalities as standard services.

* Each device should expose its capabilities to other devices.
* Other devices should be able to find available and useful services from objects around.
* The device offered services should be able to cooperate with other kinds of services (e.g. Web services) for providing more meaningful services.

Consequently, the WoO applications and components which are created by users or offered by service providers can be reused and shared among different user communities for creating innovative and personalized services.

Service exposure has several capabilities that help service/device interaction, efficient service uses with some of the benefits of service exposure as follows.

* Physical functionalities of devices can be automated which can be accessed by other objects (users or other objects).
* Automated functionalities can be provided in the form of standard services.
* Services will be easily searchable. Other devices can find and use functionalities of another object when needed.
* Traditional Web/Telecom services can interoperate with the device services seamlessly.

To make use of device-offered services, the first step is to make them machine-meaningful. Generally, devices (e.g., intelligent equipment, sensors, actuators, etc.) interconnect with each other. Their resource-constrained aspects (e.g., limited computation, communication and storage capabilities) make them more constraint than other conventional services. Moreover, devices’ availability may change more frequently. This highly dynamic information is impossible to be captured in a service description file stored in a global repository, because this leads to an unbearable level of network traffic. Therefore, some form of gateway is adopted to enable communication between these devices and external applications, and to handle the runtime discovery of devices and their services.

In consideration of these scenarios, the device service exposure can be divided into two levels: local service exposure relying on local gateways, and global service exposure which reports the basic information about the devices and their services to the global repositories via local gateways. As far as we known, the networks can be divided into LAN (Local Area Networks) and WAN (Wide Area Networks). With respect to LAN, take WSN (wireless sensor networks) for example, service exposure can be done through the transmission mode of broadcast or unicast. As for the WAN, or we can say from the viewpoint of the global network, there are many local gateways dispersed in the network while managing their own local networks using disparate technologies. Local gateway allows disparate networks to exchange information and is designed to provide an interface between non-IP networks and Internet through abstractions at the network layer. The idea behind such approach is that every connected device and its corresponding services are capable of cooperate with remain isolated and agnostic of the protocol disparity presented in their surrounding networks. Furthermore, when a user or a developer is creating a new service, he/she may not be in the same network as devices and is not necessarily familiar with the particular characteristics. This distributed device component information needs to be published in a globally-comprehensible manner to enable device capabilities to be discovered even when they are located behind local gateways, executed and composed in a unified manner.

Next figure shows device service exposure as an example for local service exposure.

Figure 3 Service exposure architecture

From the viewpoint of the global network, there are many local gateways dispersed in the network while managing their own local networks using disparate technologies. Furthermore, when a user or a developer is creating a new service, he/she may not be in the same network as devices and is not necessarily familiar with the particular characteristics. This distributed device component information needs to be published in a globally-comprehensible manner to enable device capabilities to be discovered even when they are located behind local gateways, executed and composed in a unified manner.

### Service execution platform

Grounded on the preliminary steps of syntactic and semantic service discovery and selection (as detailed in section ‎3.3.2– including its catalog of abstract components and components diagram), a part of the inner mechanic of the service execution platform will be to **semantically bridge the gap (at protocol and data level) between a service consumers and providers through the dynamic generation of “Connectors” at runtime** **(QAS20)**.

These connectors rely on the semantic meta-data provided by both consumers and providers. The “glue” often required to match the two will then be algorithmically implemented by these connectors, able to do on-the-fly data adaptation between involved parties **(QAS21)**; as informally illustrated in the following use-case diagrams (see Figure 4 and Figure 5).

In this first example use case (Figure 4), two WoO legacy systems **(QAS02, QAS22)** and a new external service are able to discover each other through the WoO registry (here represented as an abstract “CONNECT” box”). At the technical level, protocol heterogeneity between Objects will be supported, as requested from technical constraint elicited in deliverable T3.1, for both REST and SOAP. Moreover, this use case is directly linked to the following business constraint from T3.1: *“The new frameworks developed should be as much compatible as possible with existing infrastructures or easily adaptable to them”*

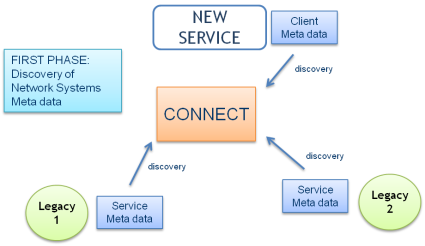


Figure 4 Service discovery - First Phase Use Case

In this second example use case (Figure 5), these same systems can be put into a transparent consumer-producer relationship, with automatic data transformation between Objects (and services), by relying on the distributed connectors that have been generated. The fact that connectors are distributed allows the distribution of the adaption work-load **(QAS02)**, avoid bottlenecks as much as possible (as opposed to a central data & protocol semantic adaptation component for the whole architecture). Moreover, generated connectors can be reused for multiple adaptations or “garbage collected” when necessary; which should allow the system to scale up/down at runtime **(QAS12)**,

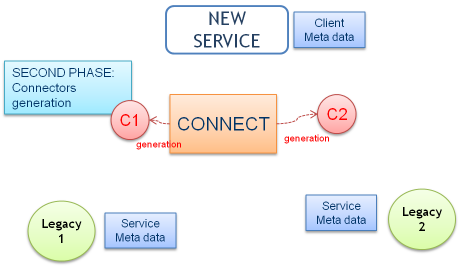


Figure 5 Service discovery - Second Phase Use Case – Connector generation

#### Service composition

Service composition in the Web of Object paradigm can be divided into two main categories: a centralized approach (Orchestration) where a monolithic application collects data and sends orders, and a distributed approach (Choreography) in which nodes provide and consumes services in a collaborative way.

#### Service Orchestration

The orchestration approach inherited from classic functional programming is centralized: a unique application, responsible for offering access and control to users, is located at a central server. It gathers all the collected data. After processing them, the central application deduces actions to perform. Orders are then transmitted to the devices present in the network. This centralized approach involves a specific traffic pattern in the network. The paths from nodes to sink are massively used, leading to congestion and high energy consumption.

#### Service Choreography

The Choreography approach advocates a distributed application logic instead of the centralized one, when application characteristics allow its implementation. This approach tries to process data and to give decision making to the nodes themselves. Each node holds a part of the application, and executes some process over the data. Small interacting parts of the application are distributed over the network. Computing is performed inside the network, and communications can often be better disseminated. When it is possible, decisions are made at node’s level, no need to send information to any central application.

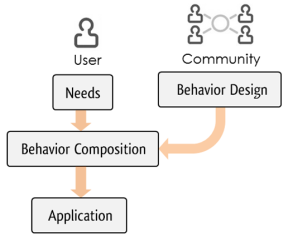
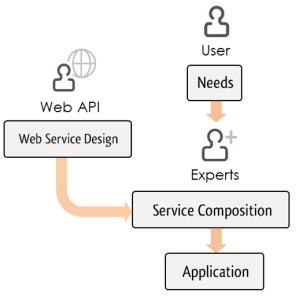


Figure 6 Service composition via Orchestration (left) or Choreography (right)

## Models

The information model described hereafter in this section defines a structure (relations, classes and attributes) about data semantic information that is managed by WoO System. This section introduces a description language to model the information exchanged in the system. This model is a general language that covers all possible situations that the objects of the WoO system may find, and it is not a representation for a specific scenario.

The information model of the Web of Objects functions as a semantic middleware that describes objects, devices and services in a Wireless Sensor and Actor Network (WSAN) using annotations of the different resources offered by the participants of the network. As a middleware, it is an abstraction layer between the WSAN components and the end user applications. This model is built on the Internet of Things Architecture (IaT-A) [1], including new features of the Web of Objects paradigm.

To describe this model, this section uses XML Schemas and OWL/RDF-S ontologies that were developed specifically to describe objects and their capabilities (ie. sensors/actuators and services) [2].

The ontological model links sensors and actuators to the services they offer. This model of semantic annotations of the available services presents some advantages, such as:

* Final users of these services are aware of every device in the network, as well as their descriptions, the features of the services, etc. from the ontology repository.
* Service composition and orchestration is possible, generating new services from simple and well-known services.
* Annotations include the specific context in which services are provided, as an accurate description of the current state of the device offering the service.
* The semantic layer adds management capabilities to the Web of Objects. This way, objects can automatically react to the current state of the network without the intervention of a human user.
* Finally, standard annotations let interoperability and integration of heterogeneous set of devices made by different manufacturers.

The modeling framework describes on the one hand the capabilities of objects according to the applications and deployment requirements in the WoO; and on the other hand the representation language of schema of respectively data or knowledge that is delivered by the objects.

The modeling components that are proposed in the scope of the project are the following:

* Data Oriented Description of Objects and Capabilities. For this purpose, we propose three general Schemas that describe respectively the input and output of the object’s services. These schemas are in line with protocols used to communicate with object and gateway services for instance, using REST, XMPP, DPWS and REST.
* Ontology Oriented Semantic Description of Objects, semantic relations and Capabilities by using Micro-Concept over RDF-S and OWL2 over RDF-S.

### Description of Objects and their capabilities

The low level description of Object (Service) capability is handled through two models. The first model is based on JSON and it allows applications to communicate directly with objects using REST. The second model is based on a XML Schema and it is a standard model that can be easily mapped to an ontology model.

This section includes a description of the model language using XML Schema. Next, the mandatory elements of the description are introduced. These elements can be enriched with new elements concerning the application and the sensors / actuators themselves.

The following schemas provide structure of the payload that is provided by Sensory objects.

|  |
| --- |
| <xs:element ref="SensorData" maxOccurs="unbounded"/><xs:sequence>  <xs:element ref="SensorData" maxOccurs="unbounded"/>  <xs:element name="SensorClass" type="xs:string"/>  <xs:element name="UUID" type="xs:string"/>  <xs:element name="timestamp" type="xs:dateTime"/>  </xs:sequence><xs:sequence>  <xs:element name="ProperyName" type="xs:string" maxOccurs="1" minOccurs="1" />  <xs:element name="ProperyValue" type="xs:double" />  <xs:element name="PropertyMetric" type="xs:string" maxOccurs="1" minOccurs="0" />  </xs:sequence></xs:element> |

To avoid the heterogeneity problems of XML descriptions of sensors and enable the interoperability of the Schemas with the ontology layer, we propose an extension Schema. This new Schema includes common sense keywords such as Metrics, Sensed Properties, Sensor Name, etc. This information can be retrieved from Web Online Taxonomy Repository such as Wordnet, DBpedia or Wikipedia. The schemas are designed to be compliant with new Microdata and RDFa Schemas that are chosen by Microsoft, Google and Yahoo for representing the Web of Object. (See Schema.org).

Table 5 Extension Schema Example for Units

|  |
| --- |
| <xs:simpleType name='allowedUnits'>      <xs:restriction base="xs:string">        <xs:enumeration value='meter'/>        <xs:enumeration value='gram'/>        <xs:enumeration value='second'/>        <xs:enumeration value='ampere'/>        <xs:enumeration value='kelvin'/>  …… ……        <xs:enumeration value='liter'/>        <xs:enumeration value='square meter'/>        <xs:enumeration value='cubic meter'/>        <xs:enumeration value='meter per second'/>        <xs:enumeration value='volt-ampere'/>        <xs:enumeration value='watt second'/>        <xs:enumeration value='percent'/>        <xs:enumeration value='enum'/>        <xs:enumeration value='lat'/>        <xs:enumeration value='lon'/>      </xs:restriction>    </xs:simpleType> |

Table 6 Extension Schema Example fo SensorClasses

|  |
| --- |
| <xs:simpleType name='SensorClass'>      <xs:restriction base="xs:string">  <xs:enumeration value='PowerMeter'/>       <xs:enumeration value='CO2Sensor'/>  <xs:enumeration value='Thermostat'/>  ….  <xs:enumeration value='Accelerometer'/>  </xs:restriction>  </xs:simpleType> |

The registration of a Device in the repository can be done using another related Schema.

We propose the following Elements

<xs:element name="ObjectRegistration">

It can contain the following sub element

<xs:element name="SensorRegistration">

<xs:element name="DeviceRegistration">

<xs:element name="ActuatorRegistration">

<xs:element name="TranseiverRegistration">

The description of the Object is given accordingly for instance in the case of a sensor:

|  |
| --- |
| <xs:complexType>  <xs:sequence>  <xs:element name="SensorClass" type="xs:string"/>  <xs:element name="ObjectUID" type="xs:string"/>  <xs:element name="DeploymentSpaceRegion" type="xs:string" maxOccurs="1" minOccurs="0"/>  <xs:element ref="SpatialCoordinate" maxOccurs="3"/>  <xs:element ref="SensorProperty" maxOccurs="unbounded"/>  <xs:element ref="Driver" maxOccurs="unbounded"/>  </xs:sequence>  </xs:complexType> |

From this point, our proposed XML Schema can be extended as needed to meet current standards such as SensorML XML Schema, OGC, Units of Measurement, etc. In addition, it can be easily mapped to an ontology in order to create a semantic registry.

**JSON Model**

This modelling language provides a general description of an object including its capabilities and output data. To this end, we add semantic annotation to JSON description [3] using a SA-REST like approach for sensor objects. We chose this approach due to the fact that offers a lightweight description language that can be easily parsed and understood by basic devices with constrained resources. The following example depicts a service registry annotation of a temperature sensor.

|  |
| --- |
| *{*  *"Entity":{*  *"Device/Actuator": {*  *"DeviceName": "Temperature Sensor 1",*  *"Id": "0014.4F01.0000.B45B",*  *"Description": "This sensor measures the environmental temperature",*  *"Manufacturer": "SunSpot Oracle",*  *"Location": {*  *"PlaceId": "Living room",*  *"Latitude": "40.6",*  *"Longitude": "-3.1"*  *},*  *"TechnologyRF": "Zigbee",*  *"BatteryLevel": "70%"*  *},*  *"Resources":{*  *"Agent":[*  *{"AgentName": "fullAgent",*  *"Services": [*  *{"ServiceName": "temperatureService",*  *"Input": "void",*  *"Output": "integer"*  *},*  *{"ServiceName": "batteryLevelService",*  *"Input": "void",*  *"Output": "integer"*  *}*  *]*  *},*  *{"AgentName": "efficientAgent",*  *"Services": [*  *{"ServiceName": "temperatureService",*  *"Input": "void",*  *"Output": "integer"*  *}*  *]*  *}*  *]*  *}*  *}*  *}* |

### Semantic Description of Objects and their capabilities

Elements of the Web of Objects use an OWL 2 compliant description model for modeling Objects and their relations using two possible representations: OWL/RDF-S or Micro-Concept RDF-S depending on the original description of objects, the targeted applications and reasoning. For instance, in case of matchmaking operations we advise using reasoning tools an OWL-RDF-S while in case of business monitoring of services orchestration we advise the use of business rules with Micro-Concept description instead of using OWL 2 mediation platforms.

The following figure shows the main concepts of the ontology providing the general description of the objects, their capabilities and their relations.

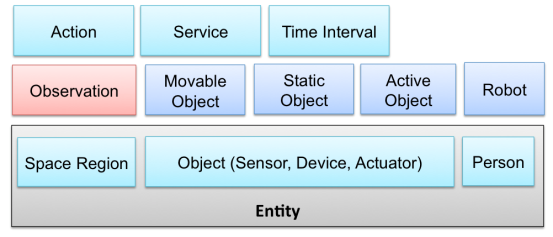


Figure 7 Main concepts for the Description of WoO Objects.

The description of an object is based on an abstract class called Entity. An entity has as synonym classes Object or Robot. It can represent a physical entity or a virtual entity. The object can inherit properties of active objects such as service, movable such as car or robot or handheld device, or static object such as sensing station.

The class Object has two subclasses:

* *Sensing Device: presents the description of a sensor*
* *Device/Actuator*: presents the description of the device or actuator. In addition, it can represent a virtual entity.
* Resources: represent the resources offered or controlled by the entities.

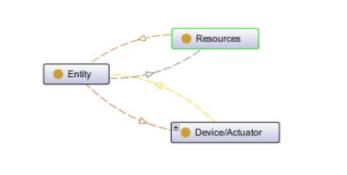
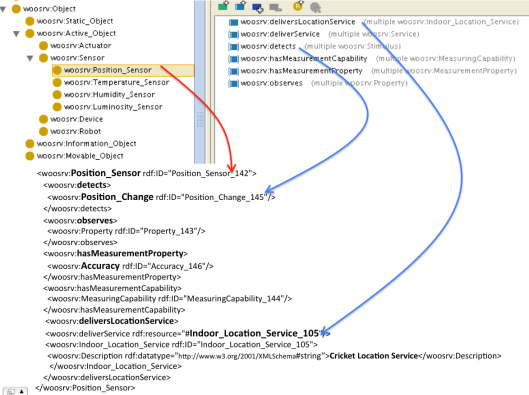


Figure 8 Partial view of Entity, Resource and Device Relation

The participants in the WoO describe (expose) the semantic capabilities of the objects according to two abstraction views. The first view allows describing the services delivered by objects for Discovery purpose, while the second view provides a technical description of the data and operations grounding in order to implement the applications that will act on or access to the services provided by the objects.



### Description of Sensing Devices

This ontology allows describing objects according to different perspectives.

The description of sensors extends the W3C SSN Ontology based onthe Stimulus-Sensor-Observation Ontology Design Pattern.

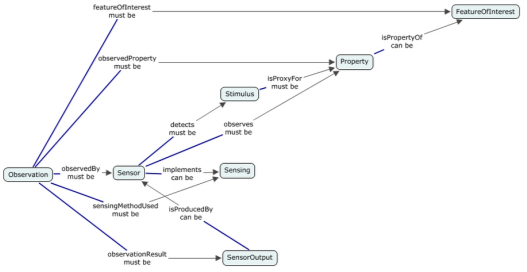


Figure 9 Stimulus-Sensor-Observation Ontology Design Pattern [3]

This approach aligns this model to the SSN domain ontology and the DOLCE DUL foundational ontology. It is beneficial on the long term to guarantee the semantic interoperability and scalability of the model with any domain applications. The second motivation behind the alignment with DOLCE is that it provides a guide for structuring application ontologies, especially regarding the taxonomic relationships.

#### Device/Actuator

The *Device/Actuator* element describes the features of a device or an actuator from a hardware perspective. This element must be published in the ontology repository to be consulted by final users. The *Device/Actuator* class is composed by the classes *DeviceName*, *Id*, *Description, Manufacturer, Location* and *TechnologyRF* and *BatteryLevel*.

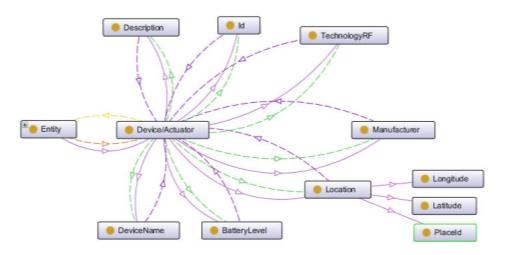


Figure 10 Device/Actuator class inside the ontology

* *Id* identifies the device uniquely. If the device is a virtual entity, the *Id* class will be equal to the attribute “virtual” and the rest of classes would appear empty.
* *DeviceName* shows the common name of this device.
* *BatteryLevel* indicates the battery level of the device or actuator.
* *Description* shows a brief description about the device presenting its characteristics.
* The class *Manufacturer* is the manufacturer of the device.
* *Location* is a class that represents where the device is. It is composed by three subclasses: *PlaceId* (is a common name of the location), *Latitude* and *Longitude*.
* *TechnologyRF* represents the technology used by the device in the communication, for example Bluetooth, Zigbee, Wi-Fi, etc.

#### Resources

The *Resources* class element represents the resources offered by the *Device/Actuator*. It has one subclass called *Agent*. An *Agent* represents a piece of software which can offer one or more services. The same *Device/Actuator* may include one or more agents (Multi agent). It must be published in the ontology repository in order to be consulted by final users.

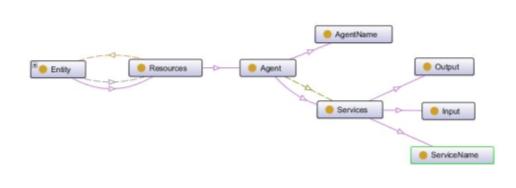
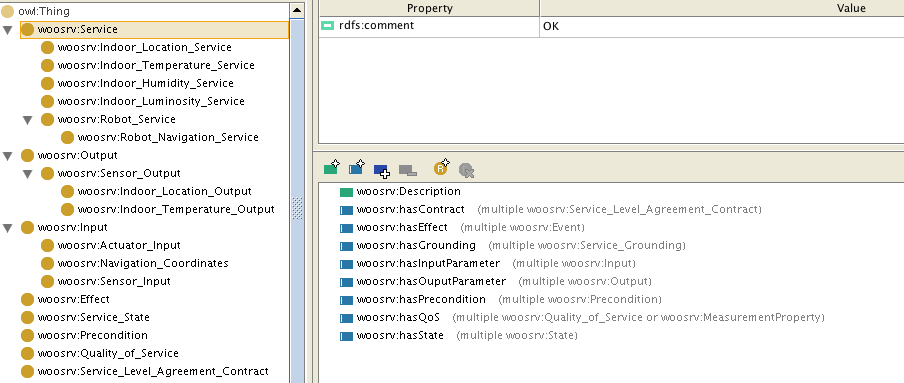


Figure 11 Resources class inside the ontology

The class *Agent* has two subclasses: *AgentName* and *Services*. The class *AgentName* represents the name of the agent. The class *Services* represents the services offered by the *Agent*. The class *Services* has three elemets: *ServiceName* indicates the name of the service, *Input* is the input information that users, processes or other objects provide to the service in order to execute the function properly, and *Output* represents the output information the service generates.

### Semantic Profile of a Service

To describe the semantic profile of object services the objects in the WoO project use an abstract ontology. This ontology can be mapped with SA-SWDl or SA-REST or With Semantic web service ontologies like OWL-S or WSMO. This model defines an ontology for services offered by the objects based on the General Concept of “Service”. A “service” is a general concept that can be specialized to describe categories of services. The service may have several descriptive properties such as Input and Output Pre-Condition and effect (Post condition), and additional non-functional properties such as Quality of Service, SLA, etc. The following figure depicts the description of the service class properties.



## Ontology description

The description has a main class called Entity. It can represent a physical entity or a virtual entity. This class has two subclasses:

* *Device/Actuator*: presents the description of the device or actuator. In addition, it can represent a virtual entity.
* Resources: represent the resources offered by the entities.

In the following figure an overview of the ontology is presented.

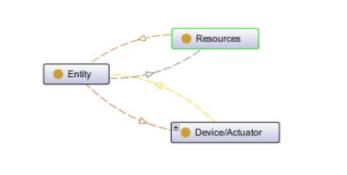


Figure 7 Ontology overview



### Device/Actuator

*Device/Actuator* describes a device or actuator in a hardware level describing its features. It must be published in the ontology repository in order to be consulted by final applications. The *Device/Actuator* class is composed by *DeviceName*, *Id*, *Description, Manufacturer, Location* and *TechnologyRF* and *BatteryLevel* classes.

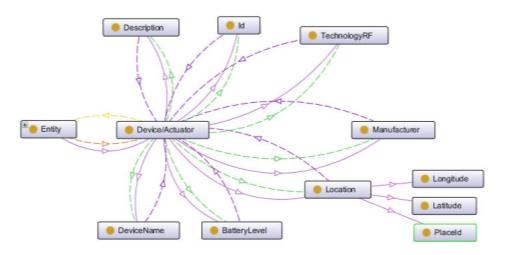


Figure 8 Device/Actuator class inside the ontology.

*DeviceName* shows the common name of this device. *Id* identifies the device uniquely. If the device is a virtual entity the *Id* class will be equal to the attribute “virtual” and the rest of classes would appear empty.

*BatteryLevel* indicates the battery level of the device or actuator. *Description* shows a brief description about the device presenting its characteristics. The class *Manufacturer* is the manufacturer of the device. *Location* is a class that represents where the device is. It is composed by three subclasses: *PlaceId* (is a common name of the location), *Latitude* and *Longitude*.

*TechnologyRF* represents the technology used by the device in the communication, for example Bluetooth, Zigbee, Wi-Fi, etc. Finally the device can indicate its battery level using the class *BatteryLevel.*

### Resources

*Resources* class represents the resources offered by the class *Device/Actuator*. It has one subclass called *Agent*. This class represents a piece of software which can offer one or more services. The same *Device/Actuator* can work with one or more agents (Multi agent). It must be published in the ontology repository in order to be consulted by final applications

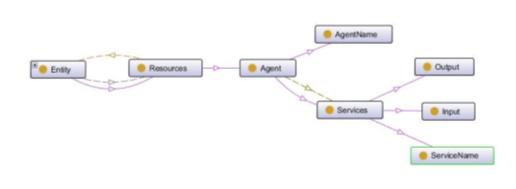


Figure 9 Resources class inside the ontology

The class *Agent* has two subclasses: *AgentName* and *Services*. The class *AgentName* represents the name of the agent. And the class *Services* are the services given by the *Agent*. At the same time the class *Services* has three parts: *ServiceName* indicates the name of the service, *Input* is the input information that user, process or other provides to the service in order to execute the function properly, and *Output* which represents the output information generates by the service.

In WoO project, the semantic annotation is done using JSON [3]. We choose this option because it is a lightweight language for devices with resource constricts. Following, a service registry annotation example of a temperature sensor is presented.

*{*

*"Entity":{*

*"Device/Actuator": {*

*"DeviceName": "Temperature Sensor 1",*

*"Id": "0014.4F01.0000.B45B",*

*"Description": "This sensor measures the environmental temperature",*

*"Manufacturer": "SunSpot Oracle",*

*"Location": {*

*"PlaceId": "Living room",*

*"Latitude": "40.6",*

*"Longitude": "-3.1"*

*},*

*"TechnologyRF": "Zigbee",*

*"BatteryLevel": "70%"*

*},*

*"Resources":{*

*"Agent":[*

*{"AgentName": "fullAgent",*

*"Services": [*

*{"ServiceName": "temperatureService",*

*"Input": "void",*

*"Output": "integer"*

*},*

*{"ServiceName": "batteryLevelService",*

*"Input": "void",*

*"Output": "integer"*

*}*

*]*

*},*

*{"AgentName": "efficientAgent",*

*"Services": [*

*{"ServiceName": "temperatureService",*

*"Input": "void",*

*"Output": "integer"*

*}*

*]*

*}*

*]*

*}*

*}*

*}*

## System lifecycles



### Service

Every service that is known to be available in the system is defined in a template that provides the means for accessing the information delivered by the devices in the case of sensors, or for interacting with the environment in the case of actuators.

The service is a way to access the resources of a device, exposing them to the system and to users. The system uses can use a peer representation of the services offered by a device to make them available to the users, or they can be directly visible from the WAN.

When a new device is connected to the WSAN, its resources are exposed by instantiation of the service templates provided by the system. The system must be aware that at a time there can be a considerable number of devices providing the same services (QAS02), providing the mechanisms to optimize the use of the system resources. For instance, instead of deploying a new service component peer at the base station for every service instantiated, there can be a general template for accessing the service, using an identification for the instantiation —e.g. device id— as a selector.

#### Execution lifecyle

A service is first instantiated when a device is plugged into the WSAN. Once instantiated, it is started, becoming active. An active service is one that can receive requests for access to information or requiring actions, according to the nature of the device.

An active service can be updated for providing new information, new access to actions, or whatever change is demanded. And of course it processes the received requests.

When a service is no longer required, for instance when a device is disconnected from the network, the service is stopped.

A depiction of this lifecycle is provided in Figure 10.



Figure 10 Activity diagram for execution lifecycle.

### Device

#### Registration

By definition a device should provide a predefined set of services according to its type, and also can provide additional services that are described in a semantic way when it is connected to the network.

There are two ways of registering a device in the Service & Device Registry. A stated in ‎3.3.1, services and devices can be registered manually before the system starts operation. For this scenario an agent that can be an authorized manager or application, accesses the Service & Device Registry directly through the management component.



Figure 11 Sequence diagram for manual registration

The other way is a dynamic registering that happens when a new device is deployed in a running system. In this case the device identifies itself to the broker component in the WSAN, exposing any additional services that it can provide, if any. Then the broker registers in its internal registry the information regarding the device and the services provided, and also populates the registration process to its pairing in the platform, that is, the management component which creates the appropriate records in the Service & Device Registry.

Also in this process are implied a gateway component, with its sink peer. Both serve as a communication bridge between the WSAN and the base station.



Figure 12 Sequence diagram for automatic registration

Another device can also provide orchestrated services, being registered as illustrated in Figure 13.



Figure 13 Sequence diagram for composed services registration

#### Un-Registration

As with the registration process, an authorized agent cans manually un-register a service and/or a device through the management component as follows:



Figure 14 Sequence diagram for manual un-registration

And when a device abandons the WSAN, it is automatically unregistered. Again, the device notifies its departure to the broker, which manages all the subsequent process.



Figure 15 Sequence Diagram for automatic un-registration

## Security



### Security requirements

From tasks T2.3 and T3.1 a set of functional and non-functional requirements, drivers and constraints have been extracted that can be applied in the design of the Reference Architecture described in this document. The following table describes the main common requirements and constraints described in D2.3 and D3.1 related to security.

|  |  |  |  |
| --- | --- | --- | --- |
| **D2.3 Common Functional Requirements** | | | |
| **ID** | | **Description** | **Requirements** |
| **WoO\_FR\_6** | | WoO platform shall enable to associate with a device access level rights concerning its access to other devices | Access Policies Management, Access Control |
| **WoO\_FR\_6** | | WoO platform shall enable to associate with a device access level rights concerning its access to other devices | Access Policies Management, Access Control |
| **WoO\_FR\_7** | | WoO platform shall enable to set access rights and policies for regulating the access from another device to the current device globally, and/or to some of its services | Access Control, Access Policies Management, Privacy Protection |
| **WoO\_FR\_12** | | WoO platform shall be able to specify the stakeholder that is the proprietary of a specified device, and the other stakeholders that have the right to use the device | Access Control, Authorization, Access Policies Management |
| **WoO\_FR\_14** | | WoO Platform shall make sure that transparency exists between different entities making use of the WoO Platform | Trust, Transparency, Fulfillment of Policies and Obligations |
| **WoO\_FR\_16** | | WoO Platform shall make sure that it remains shielded from its users | Robustness, Data and Privacy Protection |
| **WoO\_FR\_17** | | WoO Platform shall allow applications and services to access required data and cache in a secure and unified manner | Data Protection, Access Control, Privacy, Authentication |
| **D2.3 Common Non-Functional Requirements** | | | |
| **ID** | **Description** | | **Requirements** |
| **WoO\_NFR\_1** | WoO platform shall assure the access protection to a device according the access rights established in the device configuration | | Access Control, Data and Privacy Protection |
| **WoO\_NFR\_2** | WoO platform shall assure the visibility of a device and the accessibility to its services, according the access rights established in the device configuration | | Access Control, Data and Privacy Protection |
| **WoO\_NFR\_4** | WoO platform shall assure secure communication between the integrated devices | | Secure Communications, Data Protection, Verification of authenticity |
| **WoO\_NFR\_5** | WoO platform shall provide unique identifiers for the configured devices, and shall capture and employ the unique identifiers of the other integrated devices | | Data Integrity, Secure Interoperability |
| **WoO\_NFR\_6** | WoO platform shall assure data integrity for the managed WoO\_Devices-Directory. The data stored should not be tampered | | Data Integrity |
| **WoO\_NFR\_7** | WoO platform shall provide access control mechanisms to the managed WoO\_Devices-Directory | | Access Control |
| **WoO\_NFR\_9** | WoO platform shall provide logs in order to facilitate security auditing without compromising user identity and trust | | Security Auditing, Intrusion Detection, Data Integrity, Trust |
| **WoO\_NFR\_15** | WoO Platform shall make sure that user messages and commands are properly authenticated and not anonymous unless explicitly required by a use case | | Data Integrity, Privacy Management |
| **WoO\_NFR\_20** | WoO Platform shall ensure that any locally stored data is not accessible by non-authorized users and should be not tampered in anyway | | Data and Privacy Protection, Data Integrity, Access Control |
| **WoO\_NFR\_23** | WoO Platform shall take preventive measures against propagation of malicious code, viruses and other forms of malware, adware and related stuff | | Trust, Authentication, Data Protection, Creation of Security and Privacy Policies |
| **D3.1 Quality Attributes** | | | |
| **ID** | | **Description** | **Requirements** |
| **QAS01** | | The system must be prepared to add new auto-identification mechanisms/technologies | Identity Management |
| **QAS06** | | The system must provide secure communication mechanisms | Manageability |
| **QAS11** | | Physical Identification of core system devices | Manageability |
| **QAS14** | | The user profile must be secure, accessible and updatable | Data and Privacy Protection, Manageability |
| **QAS15** | | The system must be able to authenticate users to prevent anonymous and malicious access | Authentication |
| **QAS16** | | The system must be able to identify users based on the access level they are granted and provide them relevant information accordingly | Access Control |
| **D3.1 Constraints** | | | |
| **ID** | | **Description** | **Requirements** |
| **Business Constraints**  **(BC)** | | All video surveillance systems must comply with the different regulations on surveillance in public and private spaces for each country | Data Protection, Privacy, Data Integrity, Confidentiality |
| All the devices and systems that use or store identifiable personal information must comply with the current laws on the treatment of personal data, which includes the implementation of effective safeguards to preserve data privacy and integrity |

Table 5 Security requirements

### Security components

Once the main security requirements are identified, it is possible to define the components that the architecture should include to deal with them:

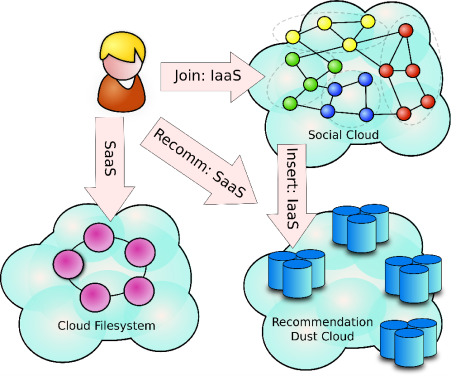
* **Authentication**: Authentication shall be required to access certain services or objects in the Web of Objects to confirm the identity and credentials of the nodes involved in the interaction, or to verify the authenticity of the messages exchanged. This process establishes a secure environment for the communication between existing and unknown new objects appearing in the network.
* **Authorization**: This component is responsible for determining the level of access that an authenticated user, device or object has to a particular resource in the Web of Objects. This task includes the management of security policies required for the secure interaction between objects in the network.
* **Identity Management**: The Identity Management deals with the creation, administration and protection of identities for the users and objects in the Web of Objects. This component determines how users and objects obtain unique identities, to consume services or other objects, and provides the necessary mechanisms to protect and limit the disclosure of personal identifiable information in the network based on defined security and privacy policies.
* **Key Exchange & Management**: This element enables secure communication between objects in the Web of Objects providing mechanisms to share keys that can be used to protect the data exchanged and facilitating the interoperability between unknown objects.
* **Trust**: The Trust component gathers and provides information of the level of trustworthiness of the objects and entities offering services in the Web of Objects, which also facilitates the interaction between objects, particularly when sensitive data is exchanged.
* **Privacy Management**: This component explores the sensitive data that objects collect from the environment and exchange in the network. The WoO paradigm aims to build a web of objects with sensors and actuators that are able to collect data from private environments such as homes, offices and cars. Hence, objects are going to manage sensitive information. This module identifies the private data, decides about the suitable policies to enforce according to the nature of the data and, if necessary, distorts, hides or joins in some way the data to protect the user.

This other table summarizes the services provided by each security component and the requirements fulfilled.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Security**  **Component** | **Services Provided** | **Requirements Covered** | **Related requirements from D2.3, D3.1** |
| **CC\_1** | **Authentication** | * Verification of identities and credentials * Verification of the authenticity of the messages exchanged | * User authentication * Data Integrity * Data origin authentication * Secure communications | WoO\_FR\_7, WoO\_FR\_12, WoO\_FR\_17, WoO\_NFR\_1, WoO\_NFR\_2, WoO\_NFR\_4, WoO\_NFR\_6, WoO\_NFR\_15, WoO\_NFR\_20, WoO\_NFR\_23,  QAS06, QAS15, BC |
| **SC\_2** | **Authorization** | * Obtain the permissions of a user/object to perform certain actions * Manage access policies | * Access Control * Data Integrity * Privacy Protection * Access Policies Management | WoO\_FR\_6,  WoO\_FR\_7,  WoO\_FR\_12,  WoO\_FR\_16,  WoO\_FR\_17,  WoO\_NFR\_1,  WoO\_NFR\_2,  WoO\_NFR\_7,  WoO\_NFR\_9,  WoO\_NFR\_20,  QAS16, BC |
| **SC\_3** | **Key Exchange & Management** | * Provision of keys for the establishment of secure channels | * Secure Communications * Secure Interoperability * Data Integrity * Confidentiality | WoO\_FR\_17,  WoO\_NFR\_4,  WoO\_NFR\_15,  QAS06, QAS14, BC |
| **SC\_4** | **Identity Management** | * Creation of unique identities taking into account security and privacy policies | * Use of unique identifiers * Privacy protection * Confidentiality * Secure Interoperability * Manageability | WoO\_NFR\_5,  WoO\_NFR\_9,  WoO\_NFR\_15,  QAS01, QAS06, QAS11, QAS14, QAS16, BC |
| **SC\_5** | **Trust** | * Collect and provide information of the trustworthiness of a certain entity | * Transparency * Creation of Security and Privacy Policies * Fulfillment of Policies and Obligations | WoO\_FR\_14,  WoO\_FR\_18,  WoO\_NFR\_9,  WoO\_NFR\_23,  QAS06, BC |
| **SC\_6** | **Privacy Management** | * Labels data as “sensitive” and protect the transmission of private data | * Data and Privacy Protection * Access control * Confidentiality | WoO\_FR\_7, WoO\_FR\_16, WoO\_NFR\_4, WoO\_NFR\_20, WoO\_NFR\_23, QAS14, BC |

Table 6 Mapping security component to requirements

### Distributed networks



Our view of the Web of Objects paradigm involves a social network of objects that is organized as a social cloud, where objects have enough communication and processing capabilities to perform complex computations and provide advanced security services. Inside this network, cluster are created and managed according to the kind of services that they provide. When objects join the network, they identify their most suitable cluster according to their description, and link to other objects in the same cluster. Several proposals in the literature create this kind of social cloud according to the interests of the users in a decentralized way.

Then, objects organize themselves and provide a joint service, in a way similar to a cloud computing architecture. In the web of objects, this cloud is organized in a dust cloud in the sense of a dust cloud. That is, each one of the objects is a virtual machine that is instantiated as needed and dismissed when the job is done. The cloud provider maintains a list of machines included in each of the N clusters of the system. In addition, another distributed network may give access to different contents, such as shared documents, commercial products, authorization rights… We will assume that, given a URL, objects are able to access a resource from this file system in a private way.

A user of these joint services must describe the kind of service that he is interested in, and he will receive a recommendation of the most suitable object inside the network that offers the service. This way of operation needs the creation of profiles that describe objects, their services and queries of the users. Additionally, queries may be issued automatically by means of the necessities of the users stated in their profiles. The protection of the private, sensitive information inside these profiles is defined in many law bodies and fundamental rights. The web of objects must not only protect the integrity of the data and prevent unauthorized access, but also it must protect the privacy and intimacy of its users.

We identify five different roles for users in our recommender system: *merchants* that provide services and their descriptions, *customers* that request services, *indexers* that reply to queries from the customers, *repositories* that provide access to databases, documents and content providers, and *intermediate nodes* that route messages from merchants and customers to indexers and repositories. In a real system, an object will play several of these roles. We will use the term *users* as a placeholder for merchants and customers when they access indexers and their specific role is not important.

#### Acquisition Network

The deployment of the sensors and acquisition data nodes of the web of objects needs a way of integrating different sensor devices with different network standards into a heterogeneous but single network. There exist previous proposals for heterogeneous Wireless sensor networks (WSNs) to autonomously intercommunicate and cooperate in order to offer a wide variety of services. Moreover, ensuring the security in this kind of networks becomes critical not only to hide sensible information to an attacker but also to avoid malicious interaction with the system. For example, a fake alarm of a surveillance system could lead to a significant waste of money and public resources. In some cases, such as eHealth networks, attacks to the network even if unintended can cause damage to the patient.

Sensor nodes are often “low-end” devices with constrained resources and hence the use of well-known but expensive security algorithms is often questionable. The availability of the gateway (as a powerful super-node which can manage security issues in a centralized manner) varies with the application space (i.e., in some applications continuity in time and space property makes the gateway not always reachable due to mobility of the patient (indoor/outdoor). Finally, the wireless broadcast medium makes the physical layer very accessible for an attacker, which can jam, inject or modify link layer packets without difficulty and which can easily compromise and spoof a sensor node.

**The Web of Objects project will have to face the deployment of a secure and heterogeneous network of devices with very different capabilities. Securing an embedded system thus generally pertains to its most important assets, notably the binary data, the logical infrastructure and the physical elements in the network. As for the first, this requires securing the logical data contents which is sensed and delivered over the network from the sensing nodes towards the data sink(s). As for the second, this requires securing the creation of the network topology, the maintenance of the network topology, the routing protocol and process as well as any associated handshaking procedures. As for the last, this requires providing adequate physical tamper resistance to ensure the integrity of data stored by the sensor nodes, including access to memory, power supply, etc.**

**To address these challenges, the project must provide some mechanisms and modules to be used by the different objects. Some of these mechanisms are:**

* **A light authentication system in the Service Discovery Protocol to allow mobile sensors with continuity in time and space properties be trusted.**
* **A light group key management algorithm allowing resource-limited and heterogeneous sensors to easily agree and exchange a key to provide confidentiality to the measured data.**
* **An intrusion detection and protection system that detects network intrusions, misuses and node failures by monitoring and analyzing activities with the purpose of identifying abnormal behaviors.**
* **A data aggregation process algorithm to minimize effect of misuse or malicious sensors, and enhance the privacy of the managed data.**
* **Propose methods for evaluating the risk of disclosure of sensitive data using semantics and the privacy protection of the exchanged data by obfuscation or anonymization methods.**

#### Key management

**One of the key technologies from the point of view of the security that the web of objects must face is the provision of a suitable key management. This way, authenticated and authorized objects may use the shared key to add security to their communication, for example, by signing their messages or encrypting their content. Hence, objects inside a group must share a common, security key. When an object joins a cluster of the web of objects, a new key must be created and distributed among the group’s members. This is a Group Key Management (GKM) scheme, which manages the changes of the shared key during the life of a group. The main challenge of a GKM scheme is the secure update and distribution of the shared key among the group’s members.**

**Hernandez-Serrano et al. aimed to a scalable key management algorithm of large groups of small, wireless devices that minimizes the re-keying cost when group members change. The group members are organized in a tree-shape structure with sub-keys in each branch. When an object joins or leaves the group, only the correspondent sub-keys inside the branches of the tree are changed. This way, the number of messages to change a key is minimized. This key management scheme does not need any central node, making it unattended and especially suitable for sensor networks. These characteristics make this scheme suitable for our needs.**

**The output of the key management scheme produces a shared key that all members of the group know. This key is updated when a new member joins the group, or an existing member leaves. We assume that entities that are not group members will not actively seek the group key. That is to say, we assume attackers may have access to the group key, but legitimate nodes that are not group members will not be interested in owning this key.**

## Components

**The following tables shows the core technologies, modules and services that have been developed in Work Package 4 for the devices and the network that supports the Web of Objects reference architecture.**

**The components are mapped to the functions, requirements & architectural drivers of the reference architecture described in this docuent,**

**For more details about these components, see deliverables 4.1, “Specifications of core technologies and software modules for the Devices and Network Technologies”, where some modules are described with detail. Other components listed here are still in progress, more details will be available on deliverable 4.2 “Final report on the WoO network and service infrastructures”.**

**The different modules, components and specifications are divided in five subsections, one per each task in WP4:** IPv6 Infrastructures, Service Framework for the Web of Objects, Resources Management, Distributed service infrastructure and Smart streaming protocol.



### IPv6 Infrastructures

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Cairo University | Module Name | Fault and delay tolerant routing support in 6LowPAN/ROLL |
| Description | | | |
| Context-aware Adaptive Routing (SCAR) is the state of the art of routing protocols developed for mobile intermittently-connected networks. According to SCAR, a node has a routing table that manages the routes to different destinations. SCAR adopts prediction techniques over context of the sensor node to compute the probability of message delivery, and hence, foresee which of the sensor neighbors are the best carriers for a given messages. Nodes exchange their contexts and their available buffer spaces to build their routing tables. According to these routing tables, nodes begin to decide to which node it delivers a message to reach the sink | | | |
| Related QAS | | | |
| QAS09, QAS12, QAS17 | | | |
| Related reference architecture components | | | |
| Managment/Communication and Devices | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Cairo University | Module Name | RPL Component |
| Description | | | |
| RPL routing protocol for Low-power and Lossy networks is the common companion of 6LowPAN / IPv6 protocols. New challenges arise in wireless networks, because of the nature of the links: in wired networks, a link is usually not oriented. On the contrary, between two wireless interfaces, it can be oriented, if one of the node has a range lower than the other, and the distance between them is located between the two ranges. Thus, RPL first discovers the network topology, in order to build a Direction Oriented Directed Acyclic Graph (DODAG). RPL is designed to allow the use of several topologies in parallel, when multiple RPL instances are defined. | | | |
| Related QAS | | | |
| QAS10, QAS12, QAS17 | | | |
| Related reference architecture components | | | |
| Managment/Communication and Devices | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Visual Tools | Module Name | Secure communications under IPv6 for video surveillance systems |
| Description | | | |
| A recommendation for the Internet Protocol Security (IPSec) defined by the Internet Engineering Task Force (IETF) to provide secure communications in IPv6 and IPv4 networks. It is a mandatory component of the IPv6 specification that must be supported in the compliant IPv6 implementations, but it can also be used to protect traffic over IPv4 connections.  It operates at the network layer and therefore it can be used to protect the packages and protocols of the transport layer, which makes this solution more flexible than others working at higher layers such as SSL, TLS and SSH. Furthermore, from the functional point of view IPSec presents other advantages over those network protocols, to name a few:   * As it is a network layer protocol, IPSec provides better encryption and authentication * Not reconfiguration is required to deploy IPSec security * IPSec provides end-to-end security without the need of a security gateway (transport mode) * Supported by different Operative Systems (i. e. Linux, Windows or MacOS) * It is an open standard, which makes interoperability between end-points easy to implement   Although generally IPSec is recommended to be used with devices which do not have very strict constraints on processing or transmission capacities (i.e. non-low-power devices or without real-time constraints) due to the overhead added, some solutions have been already developed to optimize the IPSec procotol execution, such as lightweight or compressed versions of IPSec ([LWIPSec], [CIPSec]), turning IPSec into a potential solution to secure communications in IP based sensor networks ([SNIPsec], [WSN-IPSec]). | | | |
| Related QAS | | | |
| QAS06, QAS15, QAS16 | | | |
| Related reference architecture components | | | |
| Communication: IPSec provides a secure communication channel establishing a secure environment and protecting the data exchanged  Security: IPSec provides mechanisms for authentication, authorization and verification of the data integrity | | | |

Sat

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Universidad Politécnica de Madrid – UPM | Module Name | IPv6 on Sun SPOT devices |
| Description | | | |
| The Sun SPOT is a WSN mote developed by Sun Microsystems (now Oracle Corporation), built upon the IEEE 802.15.4 standard, and also on the Squawk Java Virtual Machine.  IEEE 802.15.4 specifies the protocol for the physical and data link layers for LoWPAN, so different manufacturers have developed different network protocols. This makes difficult the deployment and integration between different solutions, so using a well-known standard on the network layer is desirable.  Here is where 6LoWPAN comes as a solution, offering a way of applying the Internet Protocol even to the smallest devices. | | | |
| Related QAS | | | |
|  | | | |
| Related reference architecture components | | | |
| Device layer in the Reference Architecture | | | |

### Service Framework for the Web of Objects

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Deimos | Module Name | Smartphone Sensors REST Framework (Gloo) |
| Description | | | |
| Gloo is a platform that exposes mobile devices sensors and capabilities as REST resources, so they can be used as smart elements in the Web of Objects and its services. This framework provides an easy mechanism to integrate data retrieved from the smartphone sensors and its owner and use them in Web of Objects services in composition with other sensors and devices. Adding the smartphone owner to the mix, humans can be also treated as a sensor in these scenarios enabling participatory sensing applications  Its main components are a proxy that acts as a gateway between services and mobile devices, the mobile agents which implement the access to mobile information, and services making use of mobile sensors and capabilities. | | | |
| Related QAS | | | |
| QAS19, QAS13, QAS20, QAS01, QAS05 | | | |
| Related reference architecture components | | | |
| WoO Backend, WoO Tools, WoO Proxy / Gategay and WoO Connected devices | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Odonata | Module Name | Embedded Service Framework |
| Description | | | |
| The Embedded Service Framework (ESF) is a redesigned, rewritten and extended version of the DPWSCore stack, which is available at http://forge.soa4d.org. The goals of this new version are:   * To bring the power of recent Web-oriented technologies to embedded devices, leveraging service-oriented and REST architectures; * To hide complexity from developers, through code generation and high-level APIs; * To support a large range of applications, from basic Web applications to complex Web services applications, featuring mechanisms such as network discovery and event publishing. * To support a wide range of platforms, from mono-threaded (or OS-less), deeply embedded devices (e.g. wireless sensors) to complex multi-threaded applications running on large devices, workstations or enterprise servers. | | | |
| Related QAS | | | |
|  | | | |
| Related reference architecture components | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Visual Tools | Module Name | Modules allowing the access to video surveillance services |
| Description | | | |
| This software module provides lightweight services for video surveillance objects which will be available through a REST paradigm to the WoO platform. | | | |
| Related QAS | | | |
| QAS04, QAS15, QAS16 | | | |
| Related reference architecture components | | | |
| Service: provides access to the resources of an object | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Visual Tools | Module Name | Module enabling the auto-configuration of video surveillance devices |
| Description | | | |
| This component allows to monitor several parameters of a video surveillance object in the WoO network (i.e.: a video recorder or a surveillance server) and reconfigure it to deal with specific limitations | | | |
| Related QAS | | | |
| QAS: QAS17 | | | |
| Related reference architecture components | | | |
| Management: this component provides to the devices the capability to be aware of changes in the system and to adapt themselves automatically to specific limitations | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | ETIC | Module Name | Web services on the WoO server or in the cloud |
| Description | | | |
| The main purpose of this module is to provide access to user profiles via REST interface. This module provides some resources, referenced by a global identifier (URI). HTTP protocol is used to access the resources, and data is exchanged in JSON format.  This module is supported by Azure Mobile Services, which offers a simple solution to build backend for mobile applications, by seamlessly integrating DB technology, along with Push Notifications and User Authentication, allowing to use different identity providers.  Apart of providing native SDKs for iOS, Android, Windows Phone, Windows Store, and .NET platforms, Azure Mobile Services integrates a built-in REST API, which makes the services accessible from any platform using HTTP protocol.  However, in order to customize the REST API, in this case, an additional layer is implemented using WCF. This allow us to customize how HTTP operations are managed and to add additional logic to the services while it is very easily integrated with Push notifications and user authentication using different identity providers. This WCF service, can be deployed as a Windows service, as a web service in an IIS server, or in the Azure cloud platform. This WCF service exposes all the REST resources available for 3rd party applications wanting to use user profiles.  Azure Mobile Services is built on top of Windows Azure, the Microsoft’s cloud platform, using it in terms of PaaS. | | | |
| Related QAS | | | |
| QAS14, QAS20 | | | |
| Related reference architecture components | | | |
| WoO Connected devices and WoO Backend | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Prodevelop/UPV | Module Name | SmartHome (REST) framework to access home appliances |
| Description | | | |
| Software module that will act as the SmartHome gateway to provide access to the appliances in a home. It will allow anyone to use those devices via REST invocations. | | | |
| Related QAS | | | |
|  | | | |
| Related reference architecture components | | | |
| WoO Backend and WoO Connected devices | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Thales Communication | Module Name | Generic Local Gateway |
| Description | | | |
| This component is a proposition of design for a local software component responsible to expose to a WAN:   * Locally connected objects and their capabilities * Locally collected data & events * Locally hosted applications (or part of applications)   This design is inspired by the “Gateway” component of FI-Ware Internet of Thing Generic Enabler (REF)  While FI-Ware considers that Data & Device management are the only functions of this gateway, we are thinking that hosting applications locally, to enable local interaction between connected objects is an important function.  In addition, we prefer to explicit functions that address security and privacy needs (authorization, authentication, anonymisation, identity management and key management).  In the context of WoO project, several components cover this generic gateway, and address specific part of it. | | | |
| Related QAS | | | |
|  | | | |
| Related reference architecture components | | | |
| WoO device, local gateway | | | |

### Resources Management

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Prodevelop/UPV | Module Name | Component RESTful specification to manage smart objects |
| Description | | | |
| Definition of a component that allow to manage smart objects and services following a RESTful pattern. This component act as a component registry, allowing to obtain the current available smart objects/services, and also enabling service discovery, service integration among gateways, among others. | | | |
| Related QAS | | | |
|  | | | |
| Related reference architecture components | | | |
| Service Registry implemented using a RESTful pattern | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Sogeti Hitech | Module Name | WS-Management over DPWS |
| Description | | | |
| Within the ambit of the Web of Objects project, the web services communication stack dedicated to devices, named DPWS, must be upgraded to be able to offer device and services management features. The management extension for the DPWS stack is based on Distributed Management Task Force (DMTF) specification named WS-Management (DSP0226, v1.1.1 released 10 Sep 2012), on the road of OSI standardization, and is working as a plug in to the DPWS stack. Indeed, it integrates all the APIs and tools needed to ensure management operations like a XML resource model, an XPath requestor API (to parse the resource model), and an embedded XML Schema validator to validate remote operations on these resources. | | | |
| Related QAS | | | |
|  | | | |
| Related reference architecture components | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Odonata | Module Name | Multi-protocol embedded resource manager |
| Description | | | |
| The multi-protocol embedded resource manager is an instantiation of ESF aimed at lightweight resource-oriented server applications. The main component of this architecture is the resource manager, which provides:   * A common API used by protocol implementations to access and modify resources. This API relies on the data binding API presented in the T4.2 description to provide a homogeneous representation of resources exchanged over the wire as XML, EXI or JSON. * A plug-in mechanism allowing developers to register resource implementations with the resource manager. | | | |
| Related QAS | | | |
|  | | | |
| Related reference architecture components | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Institut Telecom | Module Name | DPWS Gateway |
| Description | | | |
| REST interface DPWS gateway | | | |
| Related QAS | | | |
|  | | | |
| Related reference architecture components | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Smartec | Module Name | Mote Placement Optimization Tool (MPOT) |
| Description | | | |
| Different Motes hardware that help in measuring channel model of smart environments have been investigated. These motes as well as the off the shelf sensors are carefully studied for the purpose of resource management in the smart environment. Sensors energy, coverage, and connectivity are the main considerations that affect the deployment of sensors on smart environments. For the purpose of Mote Placement Optimization Tool (MPOT) deployment tool and channel model measurement , different sensors hardware is reported including their characteristic. | | | |
| Related QAS | | | |
|  | | | |
| Related reference architecture components | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Thales Communications | Module Name | Setha |
| Description | | | |
| The main abstract concepts related to the architecture of this the semantic service registry component are:   * Client: a client is a system needing to access service functionality (operation of a service called capacity). * Service: a service is a system providing functionalities (operations called capacities). * SETHA: a Thales Communications & Security middleware providing semantic search and invocation functionalities. It renders soundless the communication with concrete services deployed in the network. Access to these services is made through metadata definition of functional and non-functional aspects. Data are described by xsd schemas and semantics. * Automatic service registration in network: If a service complies with a discovery protocol supported by the server, it can detect automatically the discovery server and registers to it (QAS07). * Manual service publication: a service can be published manually in providing meta-data information to the system (SA-WSDL). | | | |
| Related QAS | | | |
| QAS03, QAS04, QAS07, QAS12, QAS20, QAS21, QAS22 | | | |
| Related reference architecture components | | | |
| Service registry, Service Discovery and Selection, Service Execution platform (partly) | | | |

### Distributed service infrastructure

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Smartec | Module Name | Sensor Deployment Software |
| Description | | | |
| This software tool deploys wireless sensors in indoor environment. The tool will have as an input, the layout of the residential/office building. Users of the tool will also be capable of laying out the furniture in different rooms as well as the material of the walls, doors and partitions. In addition, the placement optimization criterion will also be determined by the users. Examples of the criteria under consideration are the coverage, energy consumption, or cost.  This software takes care of the first step in forming a wireless sensor network, which is the deployment process. Due to the importance of this step, a large body of literature is produced to enhance and reduce the deployment process time and consumed energy. However, there are some realistic parameters that are not considered yet in the literature. We believe that taken these parameters into consideration during the deployment process will enhance the overall operation of the network as well as affect the performance of the deployment algorithms. Some of these parameters are related to the sensors; others are related to the deployment field, while some others are related to both of the sensors and the deployment field. | | | |
| Related QAS | | | |
|  | | | |
| Related architecture components | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Universidad Politécnica de Madrid | Module Name | Wireless Sensor & Actuator Network Gateway |
| Description | | | |
| For wireless sensors based on IEEE 802.15.4 and 6lowpan, it is anticipated that resource and bandwidth constraints will prevent the use of multiple protocols to access the resources: only the most efficient, based on COAP and EXI, is expected to be supported in these devices.  This limitation raises the issue of interoperability of these small devices with other participants in the Web of Objects, including:   * User applications such as Web browsers or smart phone apps; * Other devices using the full-featured Web services stack; * Management applications.   Due to the incompatibility of the protocols used in several layers, a gateway approach is proposed to perform the required conversions between the different protocols. This gateway should in general be co-located with the “edge router”, which sits between the wireless network and the IP network and performs the IPv6 header compression used in 6lowpan (Figure 47). The gateway will complete the IPv6 header translation mechanism with:   * A mapping between the HTTP and COAP protocols or, in the case of WS-Management messages, between the WS-Management addressing model and COAP. * A conversion between the XML Figure 47r JSON resource representations and the EXI representation | | | |
| Related QAS | | | |
|  | | | |
| Related architecture components | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Visual Tools | Module Name | Component to enable the discovery of smart surveillance objects in the network |
| Description | | | |
| This module allows the different smart surveillance objects to be discovered on the Web of Objects and to provide information about their properties and available services. | | | |
| Related QAS | | | |
| QAS04, QAS07 | | | |
| Related architecture components | | | |
| Communication/Service: It makes the objects discoverable, and facilitates the access and data sharing between objects | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Visual Tools | Module Name | Component for the selection of smart surveillance objects in the WoO |
| Description | | | |
| This module provides a mechanism to select dynamically the most suitable objects to compose the different smart surveillance services based on a set of predefined parameters.  Initially the parameters that will be used to the selection of a smart object will be related with the availability of the objects and the performance of the services they provide (i.e. average waiting time, accuracy of the results, etc.) but can be extended in the future to include other elements related to the quality of service, such as bandwidth consumption, the security level or the privacy policies implemented by the object. | | | |
| Related QAS | | | |
| QAS02 | | | |
| Related architecture components | | | |
| Service: It facilitates the service composition | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Elecnor Deimos | Module Name | Devices and Services registry for the Smartphone Sensors REST Framework (Gloo) |
| Description | | | |
| This component enables the registration of devices and services in order to make services available for mobile devices users and vice versa. Services need to be registered in order to be accessible for mobile devices users, and the users need to subscribe to services in order to give services access to its device sensors and capabilities. These mechanisms are not automatic, since accessing sensors and capabilities of personal devices without the user confirmation could lead to privacy issues. | | | |
| Related QAS | | | |
| QAS15, QAS16, QAS19, QAS06, 6QAS13, QAS20, QAS01, QAS05 | | | |
| Related architecture components | | | |
| WoO Backend & WoO Connected devices | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | ETIC | Module Name | Component for the discovery of WoO smart objects in different contexts |
| Description | | | |
| This module provides the appropriated mechanisms to select dynamically the most suitable objects and services for each appliaction at any context and moment | | | |
| Related QAS | | | |
| QAS06, QAS14, QAS20, QAS01 | | | |
| Related architecture components | | | |
| WoO Connected devices | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | ETIC | Module Name | NFC communication for user identification |
| Description | | | |
| A SW component that can be used to identify a user in scenarios that use NFC communication.  NDEF formatted messages are interchanged between different devices.  An initial implementation has been developed using smartphone and tablets. When the smartphone is tapped to the table, an event is fired in the tablet notifying that NFC capable device is in its range. When this happens, the tablet sends an NDEF formatted message. This message is an URI formatted message. This allows to wake up an app in the smartphone that has previously registered specific URI protocol. When the message is received on the smartphone, it asks the user for authorization to execute the app. In the case the user accepts, the smartphone will send a NDEF message to the tablet app with it ID. | | | |
| Related QAS | | | |
|  | | | |
| Related architecture components | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Sogeti HiTech | Module Name | SmartGateway / generic M2M engine |
| Description | | | |
| The huge diversity of connected devices and communication protocols in the internet of things world leads to a big issue on how we can manage and make interoperable these heterogeneous devices.  This issue is one of the major blocking point to the development of Internet of Things. Due to the incompatibility in protocols used between different devices, a Gateway/Bridge approach is proposed to perform the required conversions between the different protocols.  The smartGateway developed is usable in any business domain, and is composed of a Core (generic M2M engine) which is able to process in a generic way any events and data coming from/sent to various plug-ins available in its environment.  There is three kinds of plugins for the smartGateway:   * Plugins able to retrieve data/events from various devices/sensors/actuators, and communicates these information to the Core (the IN part in previous figure). * Plugins able to send data/events to remote clients/servers with their own protocols (OUT part). * Plugins able to add behavior to the smartGateway and to the processing of data, with additional embedded statecharts engine for example. Using behavior plugins could allow smartGateway to process internally set of data and generate new events/data based on internal behavior rules. | | | |
| Related QAS | | | |
|  | | | |
| Related architecture components | | | |
|  | | | |

### Smart streaming protocol

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Visual Tools | Module Name | Video streaming optimization based on events |
| Description | | | |
| This module allows to automatically configure the video stream depending on different events related to the surveillance service. For example: the fps of the video stream can be increased if an abnormal occupation of the store is detected. | | | |
| Related QAS | | | |
| QAS17 | | | |
| Related architecture components | | | |
| Management: this component provides to the video streaming device the capability to be aware of changes in the environment and to adapt automatically the video stream accordingly | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Visual Tools | Module Name | Video streaming security & privacy based on user profiles |
| Description | | | |
| This module will allow the modification of the video stream depending on the authorization level of the user that wants to have access to the video. This includes the necessary mechanisms to encrypt partially or totally the video stream depending on the user profile. | | | |
| Related QAS | | | |
| QAS15, QAS16, QAS06 | | | |
| Related architecture components | | | |
| Security: It provides mechanisms for authentication, authorization and privacy protection Management: with this module the video stream provided is automatically configured | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Kwangwoon University | Module Name | Adaptation manager module |
| Description | | | |
| Adaptation manager module controls rate and quality for media contents according to the quality levels. | | | |
| Related QAS | | | |
| QAS07, QAS17 | | | |
| Related architecture components | | | |
| Communication: It adaptively controls the quality of media contents according to the context information.  Management: It efficiently monitors multiple context information to offers more valuable services for the system. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Kwangwoon University | Module Name | Context manager module |
| Description | | | |
| Context manager module manages context information such as device, network, user etc | | | |
| Related QAS | | | |
| QAS07, QAS17 | | | |
| Related architecture components | | | |
| Communication: It adaptively controls the quality of media contents according to the context information. - Management: It efficiently monitors multiple context information to offers more valuable services for the system. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Partner | Kwangwoon University | Module Name | Network status monitoring module |
| Description | | | |
| Network status monitoring module monitors the network states such as available bandwidth, delay, jitter, packet loss, etc. | | | |
| Related QAS | | | |
| QAS07, QAS17 | | | |
| Related architecture components | | | |
| Communication: It adaptively controls the quality of media contents according to the context information. Management: It efficiently monitors multiple context information to offers more valuable services for the system. | | | |

# Domain specific architectures refinement

## Open Smart Neighborhood Demonstrator

The following picture depicts the deployment architecture, the artefacts that are part of the Spanish demonstrator:

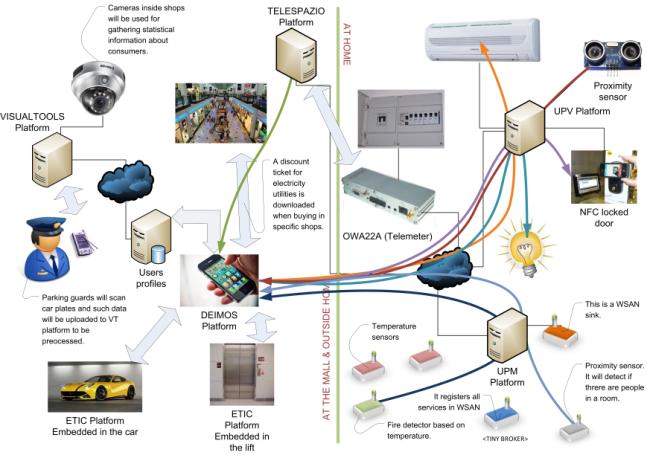


Figure 22 Deployment Architecture

In the above figure, there are clearly two different scenarios: one at home and other outside it. The resources, requirements, interactions and so forth are quite different.

At home there are several sensors, actuators, appliances and so forth. All them (their resources, capabilities, etc.) will be mapped as services and ***TELESPAZIO Platform***, ***UPV Platform***, ***DEIMOS platform***  and ***UPM Platform*** will provide them by means of either **Web Services Technolgies** or **REST Technologies**. *Telespazio Platform* will gather information from telemeters at home remotely, *UPV Platform* will manage different sensors and actuators connected to it, *DEIMOS platform* will provide access to different sensors and objects inside an smartphone (GPS locations, NFC reader, vibrator, accelerometer, etc.) and *UPM Platform* will manage sensors in a WSAN (Wireless Sensor and Actuator Network).

A system for making decisions will be deployed at home and in the smartphone.

## Egyptian Demonstrator



### Overall System Architecture



Figure 23 Egyptian Demonstration Deployment Architecture

The overall Egyptian demonstrator deployment architecture is depicted in Figure 23 comprising the Climate Control (ClimaCon). The demonstration envisions an application of the web-of-objects concepts in smart building management with concentration on HVAC and lighting equipment for both commercial and residential buildings. The system is comprised of sensor and actuator devices, gateways, end user gadgets, and back-end server(s) communicating via an IPv6-enabled wireless/wired network. The system’s main objective is to be able to manage, monitor, and customize energy consumption for both convenience and economies. The system is fundamentally based on service orchestration where the conductor responsible for application logic execution is the SBS back-end server(s).

From an architectural perspective, the sensor/actuator nodes will mostly be simple battery operated (and sometimes mains operated) devices with constraints on memory and processing capabilities and as such will mostly be incapable of supporting the SOAP/WS-\* model of web service composition. These nodes are however capable of supporting REST model. A gateway will be able to discover connected devices in its vicinity using standard service discovery services through the IPv6 network. Application-level connectivity between the gateway and sensor/actuator nodes will be over a REST-based end-to-end model using COAP protocol. A publish/subscribe model will be used to convey information from the collection points (sensors) to the sink points (gateway or application servers).



Figure 24 Service Architecture

With connectivity between the gateway and sensor/actuator nodes enabled, the gateway can proxy the nodes to rest of the world supporting DPWS and WS-\* thus enabling SOAP based services model and true web-of-objects service. This is depicted in Figure 24.

The orchestration of the system working will be the task of back-end servers. The servers will also manage user profiles, system logs, reports, and most importantly host the application logic for the intelligent control of smart building entities such as HVAC and lighting.

The Egyptian demonstrator will also include the Smart Building System (SBS) web-based application or Android App that enables the end user to do proper configuration and control as well as examine reports on various aspects of the system.

### Information Model

According to the WoO information model, the objects in the Egyptian demonstration are both the sensors and the actuators. We use REST/COAP compliant description of such heterogeneous objects and their data. REpresentational State Transfer (REST), originally developed in the context of HTTP, is a style of software architecture for distributed hypermedia systems. REST can be used to supply existing services or/and new ones in the Egyptian demonstration through a client/server architecture where the clients send requests to servers. Once the servers receive the request, they process the request and generate the appropriate response. In such a model, the basic information unit is the resource: a unique object that can be addressed through a Unique Resource Locator (URL). Hence, REST allows us to support high-level interaction between heterogeneous objects without requiring being bothered with the low-level details of the different objects. We use the WoO REST/COAP (as well as DWPS) templates for defining the services on different objects.

The Egyptian demonstration information model based on the WoO model is composed of the following main elements:

**Person:** The key player in the system which is either requesting a service (getting a temperature reading, turning ON/OFF a light bulb, etc.) and receiving a notification (e.g., a notification of sensor/actuator failure). The domain of ''Person'' is described by a set of properties:

- The profile of the user, such as name, type (administrator/user), access area within the smart building, etc.

- The user-defined profiles for modes of operation of the smart building.

- The data logs such as the energy consumption report, the fault reports, etc.

**Static Objects:** Such as walls, doors, ceilings, and other physical object in the environment in which the system in build. Actions can be defined based on the interaction between the *Person* and such objects. For example, a person entry/exit to/from a specific door to identify a room’s current occupancy, and hence, the system can set the energy consumption accordingly.

**Active Objects:** Active objects can be either:

* *Physical Objects* that delivers services or performs an operation such as Reading Data or Triggering an Action on an Object such as opening or closing a light bulb, turning up/down an air condition, get the temperature reading, etc.
* *Non-Physical Objects* which are the objects that process or store data such as the database of weather information, the database of user profiles, or the logged information database in the Egyptian demonstrator.

**Action:** The actions triggered from the interactions between objects or between human and the static objects or from the non-physical objects. Examples include:

* Turning up/down the light or the air-conditioning based on changes in the room occupancy, the ambient light/temperature.
* A user requesting a particular mode of energy consumption.

**Event:** This component of the information model represents

* The information generated by humans such as entering a room, loading a user profile, etc.
* Changes in the ambience of the system such as sun rise, sun set, etc
* The information generated by the controllable objects/sensors, such as a message exchange based on specific sensor reading, node failure, etc.

**Position 3D and Region:** Represents the coordinates and spatial relation of the region where an object is located. The concept of Region describes all the characteristics of the region. This is crucial for the control and the management of the system.

### Functional Architecture

The Egyptian demonstrator supports the following functional building blocks as defined in section 4. For each functional block, we identify the components that will be developed, revised or used as is from existing software/firmware/hardware implementation.

#### Device

The Egyptian demonstrator uses and support sensors, actuators, and gateway/router devices.

The components of the devices will be typical hardware components for each respective device. For example, a sensor node will consist of a sensing device, a micro-controller with UART-type I/O connection to external devices, and a wireless transceiver circuit. An RTOS such as Contiki is deployed to be able to support applications and protocols running on the node.

Similarly, a router/gateway device will consist of a micro-processor such as ARM9 or ARM Cortex along with wireless transceiver connecting to the sensor devices such as IEEE 802.15.4 and either Ethernet/WiFi/GSM/GPRS connectivity to enable connectivity with the global Internet.

#### Communication

The Egyptian demonstrator supports the following communication functions.

* Gateway: the gateway function will mainly be used for supporting WoO service for devices that are not directly capable of supporting that. Also, the gateway can provide bridging between different types of networks (e.g. ZigBEE and IPv6).
* Addressing: the gateway can serve as proxy for devices that are not directly addressable via IPv6.
* QoS: The architecture of the WoO will have built-in functions for support of low delay and high packet delivery ratio, as well as preserving battery life for battery-operated nodes.

On the network level, connectivity will be via a multi-hop mesh network where nodes are capable of self-organizing and forwarding packets in an efficient and fault-tolerant manner. This will be enabled by exploiting the IETF RPL routing protocol which is further enhanced by importing concepts from disruption-tolerant/delay-tolerant networking. The nodes will also be capable of discovering neighbors and obtaining IPv6 addresses automatically via the gateway(s). Efficient duty-cycling and synchronization of node transmissions/receptions will be major features of the nodes in order to preserve energy usage.

.

The following components will be used from existing Linux and Contiki implementations:

* IPv6/ZigBee stacks
* 6LowPAN implementation
* Routing/bridging function
* RPL routing in the deployed WSN to deliver sensor reading and actuator commands to and via the gateway.

To support QoS, the Egyptian demonstration will develop/enhance the following components:

* Adaptive duty cycling to preserve battery life.
* Optimized RPL implementation for enhancing packet delivery ratio and prolonged battery life.
* Neighbour discovery optimization for 6LowPAN environments as defined in RFC 6775.

#### Management

* Configuration: The system will be able to support configuration for devices, controlled equipment, user profiles, and application logic.
* Energy: The system will implement sophisticated means for energy management. Specifically using data link layer adaptive duty cycling and route optimization.
* Publish/subscribe: The devices, gateways, and servers of the system will support a publish/subscribe model for data collection and gathering.
* Fault (Limited support): Means for avoiding faulty nodes and nodes whose battery level are low or exhausted will be used. Moreover, some prediction mechanisms will be used to filter erroneous readings from erroneous sensors.
* Context (Limited support): The context of the individual devices such as the available battery capacity and the device location with respect to the gateway will be used to optimize the overall system performance.

The following components will be used:

* RESTful information access and subscribe/publish data collection as in current web services.
* RESTful based configuration management.
* Relational database management system (RDBMS) for data storage and retrieval of user data and configuration information.

The following components will be developed:

* Filtering of erroneous readings.
* Context-aware routing to avoid faulty nodes or nodes with low battery levels.
* Adaptive duty-cycling for optimized performance: for example nodes close to sink nodes (gateways) can have longer duty cycles than far nodes.

#### Service

* Service & Device Registry: This service will be supported by deploying uDPWS to enable device registry. Most of the devices will proxy that via their gateway.
* Service Discovery & Look-up: This will be enabled by a mixture of WS-\*/uDPWS at the gateway and REST based service support at the devices.
* Web of Object Services (Limited support): The Egyptian demonstrator will support limited subset of WoO services tuned to fit the expected use cases.

The following components will be used:

* Miniaturized low foot-print Web servers at devices and gateways to support RESTful architecture and publish/subscribe model.
* Service discovery protocols such as existing implementation of DPWS and Universal Plug and Play (UPnP) in current Windows/Linux and Android and other mobile platforms will be used.

The following components will be revised from existing implementations:

* uDPWS: existing implementation will be tuned to match very small footprint devices or hooks in the gateway devices will be instigated to access the device profile via RESTful architecture and present that to the rest of the world via DPWS.
* WoO: The combination of uDPWS and RESTful-based web-services models will be used to implement WoO services.

#### Security (limited/partial support)

* Initially, the Egyptian demonstrator will support user authorization and authentication for access to WoO service, device management, and system configuration. Moreover, gateway devices can block access to private networks of the WoO system from external non-authorized access. WoO devices can be accessed globally but only via the gateway device(s) which can block external non-authorized access. Initially, no attempt is made to support security at the communication level by encryption/key management. This will be done in future updates of the system once the initial concept is developed.

The following components will be used from existing implementations on Linux:

* OAuth: This is an open protocol to allow secure API authorization in a simple and standard method from desktop and web applications. OAuth is used to give users access to their data while protecting their account credentials.
* Standard Linux Kernel Firewall (iptables/netfilter) functionality to protect LAN nodes from external access. Enhanced configuration via open-source third-party firewall configuration tools such as FirewallBuilder (fwbuilder) or Shoreline will also be included.
* Network Address Translation (NAT) to link LAN devices to the Internet.

#### Application

The application architectural model of section ‎3.4 will be tuned to the different system entities such as end-user devices, sensor/actuator/gateway nodes, and application logic/database servers. The Egyptian demonstration SBS application allows the user to define personal energy consumption profiles that can be loaded to the system from a database. Also, the system has another database that records the logs of the system performance. Finally, the system allows the users to remotely configure/access the individual components of the system using their mobile device/tablet/laptop.

### Deployment Architecture in Detail

The Egyptian demonstrator is comprised of sensor and actuator nodes, gateway(s), servers (application, database, authentication/authorization) and end-user devices. The SBS application shall be deployed on end user devices and in collaboration with the Climacon servers and the building control network (sensor, actuators, and gateway) provide an overall WoO based service for smart building management. The overall deployment is depicted in Figure 23 and the device architecture shown in Figure 24.

#### Sensor Node Architecture

A typical sensor node will consist of hardware and software to provide sensing of certain phenomena and communication with a system exploiting this information via a network. It is required that sensor node have a low cost such that deployment costs will not represent an issue.

For the Egyptian demonstration purposes, we shall use readily available sensor nodes kits such as ATMEL Raven or TELOS-B. These usually have integrated sensors such as temperature sensors and can be connected via standard interfaces such UART, I2C, SPI to additional sensors. These boards also have built-in IEEE 802.15.4 transceivers. An alternative is to use versatile microcontroller boards such as Arduino equipped with shields containing the required sensors and XBee modules for IEEE 802.15.4 connectivity.

The functional architecture of the sensor nodes is detailed in **Table 9**.

**Table 9 : Sensor Nodes Functional Architecture**

|  |  |
| --- | --- |
| **Functional Block** | **Specification** |
| Device | Hardware comprised of micro-controller equipped with IEEE 802.15.4 wireless connectivity or Ethernet for mains-operated nodes. Interfaces such as I2C or UART are available for connecting the node with sensors.  Sensors for temperature, light/luminance, humidity, pressure, presence (infrared) depending on purpose of node.  Typical boards: ATMEL Raven, Telos-B, Arduino with shields for sensors and connectivity.  Firmware: Contiki RTOS (Raven and Telos-B) or barebones (Arduino via Sketches) with proper drivers for sensors and connectivity. |
| Communication | IEEE 802.15.4 transceiver for battery operated nodes.  Ethernet for mains-operated nodes.  Addressing: IPv6  Protocols: 6LowPAN, UDP/CoAP, optimized neighbour discovery for low power personal area networks as in RFC 6775.  QoS: Context-aware routing enhancement in RPL or via porting SCAR. |
| Management | * Configuration: Sensor devices will be configured for services (e.g. readings) that are exported to the system and their periodicity. * Energy: Data link layer adaptive duty cycling and route optimization for preserving energy will be deployed in the nodes. * Publish/subscribe: Sensor nodes will support a publish/subscribe model for sensor data collection and gathering via CoAP/UDP. * Context (Limited support): The context of the individual sensor nodes such as the available battery capacity and the device location with respect to the gateway will be used to optimize the overall system performance. |
| Service | The sensors nodes will mainly support RESTful service model. More capable nodes (e.g. mains-operated without memory constraints) can support uDPWS. The nodes will thus support service discovery, service registration and WoO service. |
| Security | Encryption of payload at IEEE 802.15.4 shall be configured and used by hard-coded or configuration-time encryption keys for simplicity. More security measures will be supported in the gateway nodes responsible for a set of WoO devices. |
| Application | Only simple delivery of sensor readings to subscribers via CoAP/RESTful model. |

#### Actuator Node Architecture

An actuator node is used to control other devices and equipment. It will receive control commands from the Climacon backend server(s) via the communication network and exercise the properly issue control actions to the intended equipment via its existing interfaces. For example, the actuator node can receive a command to increase the fan speed of an air-conditioner or turn on the light of a room. The actuator will have the necessary means to convey the command via the interface and using the format expected by the controlled device. The node is essentially very similar to the sensor node but instead of connecting to a sensor, will connect via a UAR/I2C/Wireless interface to the controlled equipment.

Some nodes may also act as both sensor and actuator nodes. An important architectural issue is whether or not to use long duty cycling in actuator nodes as it may not be acceptable to lose commands issued by the system to a certain actuator node.

**Table 10 : Actuator Node Functional Architecture**

|  |  |
| --- | --- |
| **Functional Block** | **Specification** |
| Device | Hardware comprised of micro-controller equipped with IEEE 802.15.4 wireless connectivity or Ethernet for mains-operated nodes. Interfaces such as I2C or UART are available for connecting the node with controlled equipment.  Typical boards: Arduino with shields for network connectivity and proper circuitry for connecting with controlled equipment.  Firmware: Arduino via Sketches with proper drivers for connectivity and control. |
| Communication | IEEE 802.15.4 transceiver for battery operated nodes.  Ethernet for mains-operated nodes.  Addressing: IPv6  Protocols: 6LowPAN, UDP/CoAP, optimized neighbour discovery for low power personal area networks as in RFC 6775.  QoS: Context-aware routing enhancement in RPL or via porting SCAR. |
| Management | * Configuration: Actuator devices will be configured for services (e.g. status/statistics) that are exported to the system and their reporting periodicity. * Energy: An important issue to decide is the duty cycling in actuator nodes and whether preserving energy is important for such type of nodes. * Publish/subscribe: Actuator nodes will support a publish/subscribe model for status/statistics data collection and gathering via CoAP/UDP. Also, they will be able to receive control actions via POST commands from the control system servers. * Context (Limited support): It is very important that routing is optimized such that commands sent to the actuators nodes are received in a timely fashion. |
| Service | The actuator nodes will mainly support RESTful service model. More capable nodes (e.g. mains-operated without memory constraints) can support uDPWS. The nodes will thus support service discovery, service registration and WoO service. |
| Security | Encryption of payload at IEEE 802.15.4 shall be configured and used by hard-coded or configuration-time encryption keys for simplicity. More security measures will be supported in the gateway nodes responsible for a set of WoO devices. |
| Application | Only simple exchange of commands with the control servers and delivery of status/statistics to subscribers via CoAP/RESTful model. |

#### Gateway Node Architecture

The gateway node plays an important role in the system architecture. The gateway will be responsible for representing WoO devices that are not capable of supporting the full SOA paradigm as envisaged by uDPWS/DPWS. Also, gateways can serve as protocol gateways between different types of networks. For example, a gateway can interconnect 6LowPAN/IPv6 and Zigbee sub-networks. Moreover (and most importantly), a gateway will typically play the role of the sink node for collecting data from a group of sensors and pre-process sensor reading before forwarding to the control servers. It will also act the source of issuing the commands to the actuator nodes as received from the control servers. A gateway can also play a pivotal role in securing connectivity of the sensor/actuator subnet to the global Internet.

**Table 11: Gateway Node Functional Architecture**

|  |  |
| --- | --- |
| **Functional Block** | **Specification** |
| Device | Hardware comprised of micro-processor equipped with Ethernet/WiFi for connection with the global Internet and IEEE 802.15.4 wireless connectivity for connecting with the sensor/actuator sub-network.  Typical boards: Raspberry PI or PCduino.  Firmware: Linux. |
| Communication | Ethernet/WiFi and IEEE 802.15.4.  Addressing: IPv6/IPv4 (for global Internet IPv4 will need to be supported)  Protocols: 6LowPAN, TCP/HTTP, UDP/CoAP, optimized neighbour discovery for low power personal area networks as in RFC 6775 within the sensor/actuator sub-network.  QoS: Within the sensor/actuator sub-network support of context-aware routing enhancement in RPL or via porting SCAR. |
| Management | * Configuration: Web-based configuration for WoO devices within the sensor/actuator sub-network. Web-based configuration for the services provided by the gateway node. |
| Service | The gateway node shall host DPWS/uDPWS for supporting WoO services for constrained devices lying within its sensor/actuator sub-network. It will communicate with these devices configuration and provisioning systems using a RESTful communication model. The gateway will thus support service discovery, service registration and WoO service. |
| Security | The external world (Internet) can only access the sensor/actuator sub-network via the gateway. The gateway will thus implement firewall functionality to protect these devices. Moreover, in coordination with the system servers, it will ensure that only properly authenticated users can access the various WoO services and resources. |
| Application | Collection of sensor readings with possible filtering and aggregation of readings coming from similar or close-by sensors. |

#### Server Architecture

The Climacon system will comprise a variety of servers that will enable its proper operation. These servers can reside on the same physical machine or run on separate machines or run as distinct virtual machines on the same physical server. The exact configuration is not relevant, so we focus here on the functions contributed by the servers in the system. We envision servers providing the following functions:

* Application logic: this server is responsible for collecting the sensor readings and based on configuration and user profiles, execute actions that will achieve the targets of the system.
* Database management: a DBMS will be used to store readings, user profiles, and energy usage and other system reports.
* Authentication/Authorization: This function will be used to properly authenticate and authorize users to have access to multiple levels of authority to access various WoO services.

**Table 12 : Server Functional Architecture**

|  |  |
| --- | --- |
| **Functional Block** | **Specification** |
| Device | A PC/desktop with proper configuration.  OS: Linux. |
| Communication | Ethernet/WiFi.  Addressing: IPv6/IPv4 (for global Internet IPv4 will need to be supported)  Protocols: TCP/HTTP, UDP/CoAP |
| Management | Configuration: Web-based configuration for all services. |
| Service | No explicit support for WoO services. However, the servers will be fully RESTful-based. |
| Security | Provide mechanism for proper user authorization and authentication. This information will be dynamically conveyed to the gateway nodes. |
| Application | Logic and intelligence for execution of the system control functionality.  DBMS  Authorization and authentication services (oAuth). |

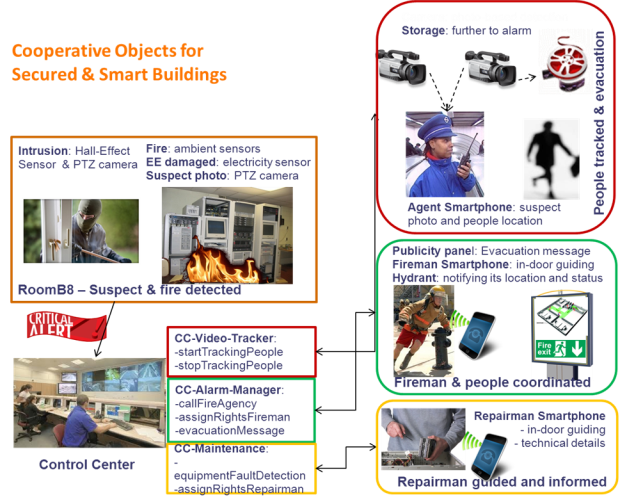
#### End-user Device Architecture

An end user device will be a smartphone, tablet, or PC/laptop that is executing the SBS application. For demo purposes, we will focus on Android phones/tablets that are equipped with the SBS App that can manage all aspects of the system.

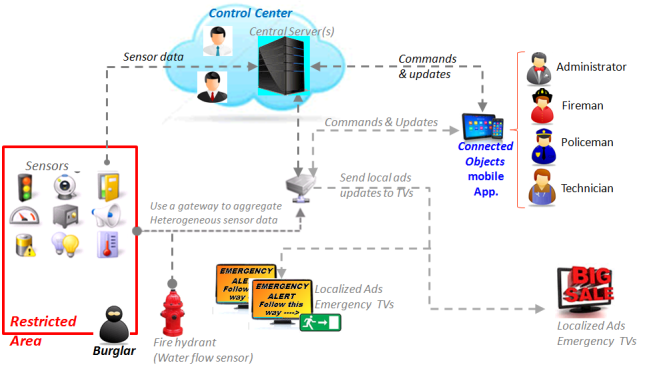
**Table 13 : End-user Device Functional Architecture**

|  |  |
| --- | --- |
| **Functional Block** | **Specification** |
| Device | Smart Phone or Tablet with WiFi/3G connection.    OS: Android. |
| Communication | WiFi/3G/GPRS.  Addressing: IPv6/IPv4.  Protocols: 6LowPAN, TCP/HTTP, UDP/CoAP. |
| Management | No specific management function for the end user device is foreseen in the architecture. |
| Service | The uDPWS will be supported for service discovery and registration and exploring WoO services. |
| Security | Proper user authorization and authentication to access the Climacon system. |
| Application | The end-user device will run the SBS supporting the defined use cases defined in WoO deliverable D2.2 Definition and refinement of use cases. |

## French Demonstrator

The following schema depicts the functional use case related to French demonstrator: Cooperative objects for Secured & Smart Buildings. It has two complementary focuses: *Building Security* (through which intrusion alarm is emitted, security camera controlled and security agents dispatched) and *Infrastructure Management* (that deals with infrastructure-level consequences of a security breach – such as automatically opening doors)

To achieve this functional goal, the following schema depicts the related overview architecture for this French demonstrator:



In a building, a restricted area is defined, with several heterogeneous and autonomous wired or wireless sensors, actuators, appliances. All of them are sending events or data on their own way, using their own protocol, either directly to a remote Control Center, or through a “smartGateway” which is able to understand one of them and to route data to the right destination devices (Control Center, remote mobile applications on smartphones or tablets, remote TV screens…).

The Sensors are heterogeneous (Zigbee, ZWave, USB…) and are used as is, out of the box, or provided by partners such as ***CNRS/Thales*** (for cameras).

The sensors data/events are:

1. Either routed directly to remote heterogeneous clients such as :

- A remote Control Center (including BeC3 server from ***University Marne-La-Vallée*** used for Infrastructure Management in the demonstrator, as well as dedicated basic Building Security Control Center provided by **Thales**)*,*

- ThingsChat remote client from***Université Paris Est,***

- Or specific connected mobile applications on tablets or smartphones.

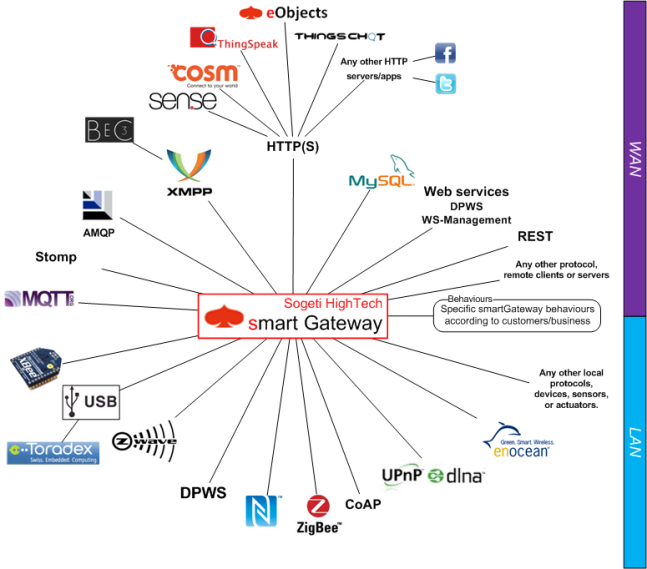
1. Or sent to the ***Sogeti High Tech*** smartGateway which is able to

- Route these data to specific remote servers,

- Transform these data on the fly (e.g. to present these sensors as XML syntaxic description through WS-Management protocol, which is a management a plugin to **Odonata** web services DPWS stack).

- Aggregate these data on the fly to generate new higher level data

- Act according to a specific behavior thanks to an embedded Sate Machine to generate new higher-level events / data before sending them to remote clients.



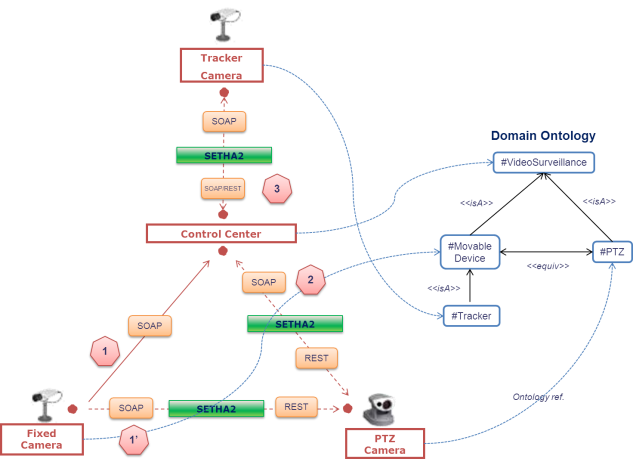
Overview of Sogeti HighTech smartGateway

From a demonstration scenario point of view, the following two main steps will happen (partly in sequence).

**Step 1: Intrusion detection with interaction with humans such as controllers or security agents for immediate response**. Building Security mechanisms and objects are used; intrusion detection is forwarded to other components (e.g. to launch step 2)

* At this level, the main actors are the Security Cameras (PTZ camera, fixed camera, Tracker camera), Building Security Control Center, Security Controllers and Security Agents.
* Communication happens mostly on a peer-to-peer basis between objects (e.g. fixed camera controlling the PTZ camera), illustrating the decentralized nature of WoO architectures.

The following figure illustrates the communication between the various Objects (as services) in step 1, as well as the attached semantic meta-data from a domain ontology extract (e.g. PTZ Camera Object that provides an ontology reference to the *PTZ* concept):



Overview of Intrusion detection and interaction with security Objects

Based, on the previous figure, the following sub-steps can be devised for the scenario:

* **Step 1.1**: Fixed Camera (FC) detects intruder at the door and sends an alarm to Control Center (C2). C2 displays video stream from FC
* **Step 1.1’**: FC requests an additional #MovableDevice capacity that can film the door, and SETHA2 dynamically matches to available equivalent #PTZ – FC moves the Pan-Tilt-Zoom Camera (PTZ) to the door
* **Step 1.2:** C2 requests #VideoSurveillance capacity, and SETHA2 dynamically matches to available compatible #PTZ – C2 displays video stream from PTZ
* **Step 1.3:** C2 requests a Tracker Camera using #Tracker ontology concept. The Tracker will be used to count the number of person in the field of view and to detect particular behavior patterns from people in the building (e.g. people walking in the wrong direction following emergency exit request) and take appropriate decisions at the C2 level (e.g. dispatching security agents).

**Step 2: Infrastructure Management activities launched following intrusion detection.**

* Communication happens mostly on a gateway-basis, illustrating the support of highly heterogeneous devices/Objects in WoO.

### Information Model

According to the WoO information model, the objects in the French demonstration are both the sensors and the actuators.

Multiple description formats and access protocols for heterogeneous objects and associated data are supported:

* For the Building Security part of the scenario, WS-\* and REST stacks are used. As such, descriptions and access are provided through REST, [SA-]WSDL and DPWS (depending on the actual sensor or actuator)
* For the Infrastructure Management part of the scenario REST, HTTP, DPWS and XMPP stacks are used.

The French demonstration information model based on the WoO model is composed of the following main elements:

**Person:** such as intruders, security agents, controllers, administrators, firemen, policemen, technicians.

**Active Objects:** active objects can be either:

* *Physical Objects* that delivers services or performs actions on command or autonomously (e.g. Pan-Tilt-Zoom (PTZ) security cameras, door sensors, magnetic door actuators)
* *Non-Physical Objects* which are the objects that process or store data (e.g. Control Center, databases).

**Action:** actions triggered from the interactions between objects or between persons and objects. Some actions will be dynamically defined (e.g. trough Bec3 choreographies) or hard-coded (e.g. as is the case with the legacy-behaving Building Security Control Center)

**Event:** all events defined above. They include sensors data/events either routed directly to remote heterogeneous clients or sent to the ***Sogeti High Tech*** smartGateway.

### Functional Architecture

The French demonstrator supports the following functional building blocks as defined in section 4. For each functional block, we identify the components that will be developed, revised or used as is from existing software/firmware/hardware implementation.



#### Devices

The French demonstrator uses and support sensors, actuators, and gateway/router devices.

Devices will rely on typical hardware components. For example, a sensor node will consist of a sensing device, a micro-controller with UART-type I/O connection to external devices, and a wireless transceiver circuit.

Other devices, such as the PTZ camera provided by Thales are composed of the camera itself (as is commonly available on the marketplace) and a small computer that stores and execute the business logic alongside. With further studies out of the scope of the WoO demonstrator, this business logic could be embedded on smaller devices.

Tracker cameras also are composed of the camera itself and a small computer used for data processing. This whole object provides a web service to the building infrastructure. For his integration in the demonstrator, the service is WS-\* compliant and enable also REST and SEAP service. The control center can send a message to the object to choose the type of information it wants to be processed (counting or raising event). The service of raising an alarm of abnormal event when somebody is in the wrong direction can be received by a client if subscribe during the step of discovering. The tracker camera also provides a service of getting the number of person in the field of view. This information is sent after receiving a request from any object in the network.

#### Communication

The French demonstrator supports the following communication modes:

* Gateway: the Sogeti HighTech Smart Gateway function will be used for dealing with Objects communication protocols heterogeneity and indirect Object addressing. The gateway can serve as a proxy for devices that are not directly addressable via IPv6.
* Peer-to-peer: direct communication between WoO Objects. This is illustrated in the Building Security step by the direct command of the PTZ camera by message exchange with the fixed camera. This communication mode also support indirect Object addressing: this is the case where semantic information is used by an object consumer to find such an object through the SETHA component provided by Thales (e.g. the semantic description of a PTZ camera as used by the Fixed camera).

#### Management

* Configuration: The system will be able to support configuration for devices, controlled equipments, user profiles, and application logic.
* Publish/subscribe: Some Objects and the gateway will support a publish/subscribe model for data collection and gathering.

#### Services

The combination of DPWS, REST and WS-\* compliant services will be used to implement WoO Objects. It will illustrate the seamlessness integration of multi-protocol Objects in the WoO platform.

* For Building Security:
  + Fixed Camera service: WS-\* compliant SOAP service with semantic meta data
  + PTZ Camera service: REST service with semantic meta data
  + Control Center service: WS-\* compliant SOAP service with semantic meta data
  + Tracker Camera service: REST and/or SOAP service with semantic meta data. It is also WS-\* compliant
* For Infrastructure Management:
  + TBD

#### Application

The application architectural model will be tuned to the different system entities such as end-user devices, sensor/actuator/gateway nodes, and application logic/database servers.

# Conclusion

This deliverable presents the first version of Web-of Objects architecture. It defines the main functions offered by WoO system, and introduces proposed models. The proposed architecture is technology agnostic and is dedicated to define functions in use in Web of Objects, propose components carrying these functions according to the models proposed in WoO D3.2, propose WoO system lifecycles and security issues. The components description is a short description referencing to more complete implementations in WP4 and WP5.

Domain specific demonstrator’s architecture and selected technologies are also described in this document.

More specifically, this document describes the different layers from the devices to the application, namely devices, communication, management, service, security and application, by exposing the services provided by each object, how they are registered, discovered, exposed and executed. A specific focus is made on the models of the objects, especially using domain-free, semantic technologies. The whole lifecycle of the WoO systems is then exposed as well as the security management.

This overall architecture is to be read as a general overview before some specific implementation details and demonstration don in the subsequent work packages: WP4, WP5 and WP6.