Object (B)logging: a Semantic-Based Self-Description for Things Networks in Pervasive Contexts

Politecnico di Bari

Department of Electrical and Information Engineering
ELECTRICAL AND INFORMATION ENGINEERING
Ph.D. Program
SSD: ING-INF/05

Final Dissertation

Object (B)logging: a Semantic-Based Self-Description for Things Networks in Pervasive Contexts

---

by
Eliana Bove :

Supervisor:

Prof. Michele Ruta

Coordinator of Ph.D Program:
Prof. Vittorio Passaro

XXIX cycle, 2014-2016

Department of Electrical and Information Engineering
ELECTRICAL AND INFORMATION ENGINEERING
Ph.D. Program
SSD: ING-INF/05

Final Dissertation

Object (B)logging: a Semantic-Based Self-Description for Things Networks in Pervasive Contexts

---

by
Eliana Bove :

_____
Firma leggibile e per esteso

Referees:

Prof. Michael Sheng

Prof. Marco Sacco

Supervisor:

Prof. Michele Ruta

_____*firma*

Coordinator of Ph.D Program:

Prof. Vittorio Passaro

_____*firma*

XXIX cycle, 2014-2016

Contents

List of Figures

List of Tables

Abstract

This thesis proposes *Object (b)logging*, a novel general approach for a Semantic Web of Things, based on an evolution of conventional Web of Things paradigms and introducing ubiquitous Knowledge Base KB models in order to associate semantic annotations to real-world objects and events. Object (b)logging represents the capability of an object to describe itself and its context in a fully automated fashion starting from raw environmental data collected by sensors.

The overall goal is to define a knowledge-based framework for high-level information representation, knowledge discovery, allotment and sharing in distributed scenarios populated by smart objects. Smart object is an intelligent agent equipped with embedded sensors, actuators, communication interfaces, computation and storage. Several heterogeneous micro-devices cooperate to connote and modify appropriately the state of the surrounding environment.

By leveraging the integration of standard machine learning techniques with non-standard semantic-based reasoning services, the dissertation defines software architectures and methods to enable efficient automated context annotation on resource-constrained mobile computing devices and to progressively improve produced descriptions during the object's lifetime. Throughout the objects lifetime, the acquired knowledge is exposed to the outside world as in a blog: to achieve this, the proposal includes a layered architecture built on a publish/subscribe Message Oriented Middleware.

A novel collaborative and distributed information sharing approach is enabled in pervasive computing scenarios featured by volatile nodes interacting in an opportunistic fashion. It is based on the *ubiquitous KB (u-KB)* paradigm, which allows to manage in a decentralized way knowledge scattered on several nodes within a network. Semantic matchmaking is exploited to support dynamic and flexible knowledge discovery.

The various elements of the framework were implemented in suitable prototypical testbeds and experimental analysis was carried out with reference to selected case studies. Results indicate the feasibility and usefulness of the envisioned approach.

Chapter 1

Introduction

The *Internet of Things* (IoT) vision is increasingly enabled by the miniaturization of microelectronic devices, enabling the deployment of a relatively large number of heterogeneous micro-components capable of storing and exchanging not-negligible amounts of information. A significant limitation of current IoT lies in a limited compatibility of devices and software stacks from different manufacturers. This forces the design of single-purpose object networks and solutions, impairing a wider usefulness of the technology. The relevance and the interoperability of the IoT could be enhanced by embedding semantically rich and easily accessible information into the physical world. This visionhas been called *Semantic Web of Things* (SWoT) [62] as the join between the Semantic Web and the Internet of Things paradigms. The SWoT improves intelligence of embedded objects and autonomic information management in pervasive contexts, in order to better support user activities and provide general-purpose innovative services.

In pervasive computing , information is scattered in a given environment in the form of atoms which deeply permeate the context [23]. Heterogeneous data streams are continuously retrieved and locally processed by mobile adhoc networks of *smart objects* dipped in the environment, in order to detect events of interest in observed areas. A smart object [5] is an intelligent software agent acting on a device equipped with embedded sensors, actuators, communication ports as well as (usually constrained) computation, storage and energy resources. Each smart object describes itself and the context where it operates toward a variety of external devices and IoT applications. However, the existing approaches defined for smart objects use data mining methods and adopt architectures designed for single applications. In order to improve flexibility and interoperability, Semantic Web standard technologies [11] can be adopted for rich and unambiguous semantic-based information exchange. The essential benefit for smart entities concerns the integration of

Knowledge Representation and Reasoning (KRR) capabilities into objects to automatically extract and process implicit useful information starting from explicit event and context detection.

This work proposes a novel semantic-based framework is proposed for high-level knowledge representation, discovery and sharing within smart object networks in the Semantic Web of Things. By leveraging the integration of standard supervised Machine Learning (ML) techniques with non-standard semantic-based inference services [61] on annotations in Semantic Web languages, smart objects become able to annotate in a fully automatic way the context they are in, continuously enriching their basic descriptive core according to events and phenomena they detect and exposing their description toward the rest of the world in a self-contained fashion as in a blog. Identification and sensing information are expressed in Web Ontology Language (OWL 2) annotations [77] via a semantic-based evolution of standard *k Nearest Neighbors* (k-NN) ML algorithm.

The proposed approach relies on both ideas and technologies of distributed knowledge-based systems [62], whose individuals (assertional knowledge) are physically tied to locations and objects disseminated in a given environment, without centralized coordination. For knowledge sharing in smart object networks, the proposed framework exploits the publish/subscribe (pub/sub) Message-Oriented Middleware (MOM) architectural model for interconnecting distributed components. In order to enable a dynamic knowledge discovery, the proposal adds two semantic enhancement layers. In detail, the topmost layer leverages standard and non-standard inference services for semantic matchmaking to support dynamic logic-based discovery of topics associated to services/resources characterized by semantic annotations. This enables a fine-grained categorization and ranking of resources matching a request by integrating an optimized Description Logic (DL) reasoner for mobile and embedded devices [70]. The middle layer is a distributed collaborative protocol used to collect fragments of the vocabulary *ontology* needed to annotate resources disseminated among the devices in an environment. It is responsible for rebuilding a minimal ontology core needed in order to support inference procedures on a particular set of semantic annotations. As ontologies can be large and Semantic Web languages use the verbose XML syntax, both compression and ontology partitioning are needed. The thesis defines a novel scheme for reconstructing partitioned ontologies. It seeks a practical trade-off between the size of individual ontology fragments managed by devices and the number of message exchanges required for on-the-fly reassembly. This layer implements a *ubiquitous KB* (u-KB) [62] model, where a node of the distributed system endowed with a reasoner fetches on the fly all and only the KB parts required for the current inference problem. Finally,

the lowest layer is an off-the-shelf message-oriented middleware based on the publish-subscribe model. It provides services for real-time data distribution among loosely-coupled components to support functionalities of the higher layers. The main goal of the overall dissertation is to achieve objects co-operation for triggering actions or making interventions on the environment according to the detected context.

The proposed approach results in a general-purpose, cross-domain semantic-based context mining, knowledge discovery and sharing facility among pervasive smart devices. In order to evaluate the usefulness of the proposed framework in a real scenario, the theoretical approach has been implemented in a prototypical testbed exploiting several robotic platforms. The thesis makes every object involved in the scenario able to summarize the information gathered via its sensing interfaces into a semantically annotated description of the environment and relevant entities in it. Furthermore, robots can interact and communicate with each other by leveraging the proposed knowledge sharing approach which integrates BEE Data Distribution System (BEE-DDS) [1].

A prototype was implemented and tested in a simulation campaign, basically devoted to assess its feasibility, correctness and sustainability. Performance evaluation was carried out with reference to selected case studies, evidencing strengths and weaknesses of the approach.

The remainder of this thesis is organized as in what follows. Chapter 2 recalls the technological background for the research and provides a survey of related work. Chapter 3 discusses the proposed framework in detail. Chapter 4 describes the implemented software protypes and presents experimental results by considering illustrative case studies to allow a better understanding of the dissertation. Finally, Chapter 5 closes the thesis.

[1] BEE Data Distribuition System, `http://sine.ni.com/nips/cds/view/p/lang/it/nid/211025`

Chapter 2

Background

This chapter presents an overview of the state of the art regarding the Semantic Web of Things technologies adopted in ubiquitous and pervasive contexts where several heterogeneous mobile micro-devices sense their physical environment and adapt their behavior accordingly. Afterwards, generalities are recalled about the tools adopted in the proposed approach: Knowledge-based systems, Machine Learning algorithms, Description Logics and semantic-based matchmaking, and inter-node communication platforms.

2.1 Pervasive computing: context and smart objects

Pervasive computing [66] is based on the growing adoption of mobile devices. The father of the ubiquitous computing philosophy is Mark Weiser, scientific director of technological research at Xerox PARK. In [79], he announced a change in the way of conceiving the automatic information processing and described scenarios where computers are ubiquitous and increasingly become part of everyday life. The main goal of pervasive computing is to create ambient intelligence where network devices embedded in the environment cooperatively and autonomously collect, process and transport information, in order to adapt to the associated context and people's activities, providing unobtrusive connectivity and services all the time. The key features characterizing the pervasive computing are:

- *ubiquity*: the system interaction is available everywhere;
- *embedding*: electronic computing components are integrated into everyday objects in order to transform them into smart objects;

- *miniaturization*: the hardware units are manufactured as ever smaller mechanical, optical and electronic devices;
- *context-awareness*: software architecture for pervasive scenarios must to be able to perceive the context;
- *self-adaptation*: the system adapt its behavior accordingly to the detected context.

The definition of context is provided by Dey and Abowd in [2]:
"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."

Context adaptation stresses the challenge of making software architecture modular and configurable at run-time for both functional and non-functional logic components in order to confer flexibility in all the possible operating contexts. The capability of creating modular and reconfigurable software at runtime dramatically simplifies the design of pervasive applications.

Context representation is a fundamental aspect in pervasive computer systems where a bottom-up approach is exploited starting from sensor data representing aspects of the physical environment [25]. Data streams coming from heterogeneous sensors are inherently imprecise and inaccurate, noisy, with different sampling rates and complex - often implicit - correlations with each other.

In this work a combination of Semantic Web technologies is used to support storage, representation, exchange, manipulation and programming with sensor data. [81] highlights the benefits arising from Semantic Web technologies in pervasive computing. In particular, they facilitate the unambiguous sharing and understanding of domain knowledge across heterogeneous and distributed systems. The most relevant and widely used model for providing high-level abstraction of sensor data is the World Wide Web Consortium (W3C) Semantic Sensor Networks (SSN) ontology [20]. It is based on an extension and integration of different works in literature: CESN ontology [16], OntoSensor (a prototype sensor knowledge repository) [59], a service-oriented ontology for Wireless Sensor Networks [43] and ontologies for sensor networks describing the topology, network settings, sensor properties, dataflow and sensor network performance [41, 27]

Recent challenges in pervasive scenarios concern connecting heterogeneous domains with each other and interpreting sensor data in an interoperable manner. Four types of interoperability are defined in [9]:

- technical interoperability, associated with hardware/software components;
- syntactical interoperability, related to data formats. This aspect underlines the need to agree on common well-defined syntax, encoding and vocabularies in order to describe data;
- semantic interoperability, associated with the machine interpretation and meaning of the content;
- organizational interoperability, related to heterogeneity of the different infrastructures involved.

Several application areas adopt pervasive computing paradigms in order to connect the physical world with the virtual world. They include: tracking goods along the entire supply chain, analyzing traffic through assistance systems, averting and fighting external threats in military sector, health monitoring for the elderly, configuring and controlling decentralized production systems automatically, selling products in business transactions, exchanging information among home devices, appliances and subsystems (smart homes), environmental monitoring and many others.

For all these application fields, the purpose is to meet the claim of *"everything, always, everywhere"* for data processing and dissemination through the ubiquity of embedded processors, computers, sensors and digital communication technologies as inexpensive commodities permeating the environment.

Pervasive computing environments are permeated and flooded by intelligent entities called *smart objects*. A smart object [76] is an intelligent software agent acting on behalf of a device, equipped with embedded sensors, actuators, communication ports as well as computation and storage facilities. It is able to retrieve a set of external data streams and can adapt itself accordingly and/or act on the surrounding environment in order to modify it. An agent is defined intelligent if it is capable to make rational decisions, *i.e.*, it is designed to maximize a metric of utility according to its objectives [58]. Figure 2.1 illustrates the general scheme underlying the smart object self-description mechanism.

Three basic kinds of information are managed:

- **Environmental data**: a description of the context where the object operates, according to data collected through a sensor set;

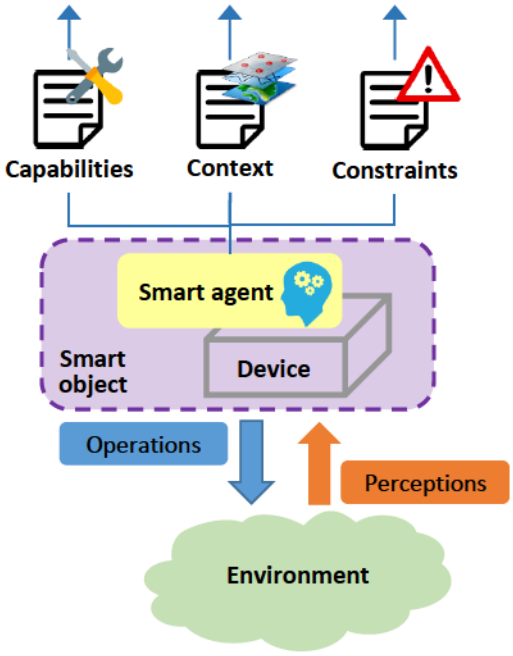


Figure 2.1: Smart object descriptions

- **Capabilities**: smart object sensing and actuation features;

- **Constraints**: limits and features imposed by other entities and influencing the smart object's behavior.

In the proposed object (b)logging approach, this information is semantically annotated with reference to an ontology, *i.e.*, an appropriate vocabulary used to provide a shared conceptualization of a certain domain of interest. Descriptions are based on the descriptive core of a smart object; throughout the object's lifetime, this semantic endowment is progressively enriched and completed so that it can be exposed to the outside world as in a blog. Via blog entries ("posts"), objects can interact at the application layer with humans and other enitites in order to create an efficient active social network where all nodes are able to exchange information, discover new services, start new acquaintances, connect to external services, share knowledge, exploit other objects' capabilities and collaborate toward a common goal [5].

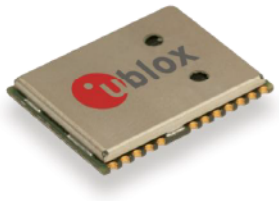


Figure 2.2: GPS NEO-6G u-blox

2.2 Hardware platforms

Almost all pervasive computing applications leverage various hardware technologies in order to realize intelligent and interactive environments. Microeletronics is essential for mobility and embeddedness in this area. The following subsections discuss the array of technological hardware solutions adopted in this work to build pervasive computing frameworks.

2.2.1 Sensors

The main characteristics of smart objects operating into pervasive contexts is to capture and analyze the surrounding environment by exploiting sensing devices. Object sensing capabilities quantitatively and qualitatively register the features of the environment considering different aspects, ranging from positioning and orientation, localization, ambient detection, to many others. Embedded sensor development includes several challenges: size and weight reduction, decrease in power consumption, the development of lower-cost production technologies, integration into complex semiconductor systems and increase in sensor performance and reliability. New approaches have been defined in nanotechnology for the realization of smaller and more sensitive sensor elements. Different types of sensors needed to acquire data are presented below.

Position and orientation sensors

Position and orientation sensors are used for perceiving the location occupied in space and the inertial alignment of a specific entity.
GPS (Global Positioning System): used to determine the absolute position on Earth of the device. It processes radiosignals received from a constellation of 24 satellites placed into orbit by the U.S. Department of Defense

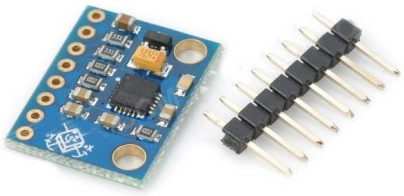


Figure 2.3: MPU-6050 (GY-521) contains both a 3-Axis Gyroscope and a 3-Axis accelerometer

[1] on which the system is based and carries out the necessary calculations in order to deduce its location. A GPS module[2] for microcontrollers and microcomputers is shown in Figure 2.2.
Accelerometer: measures the inertia of a mass when it is subjected to an acceleration on one of the three axes.
Gyroscope: allows to maintain the axis of rotation in a fixed position according to the conservation of angular momentum and is useful for measuring or maintaining orientation of the device.
Usually accelerometer and gyroscope are assembled in a single module in order to reduce costs by improving the compactness. The board[3] shown in Figure 2.3 integrates both sensors to an Arduino microcontroller.

Remote sensing

The devices able to graphically map the characteristics of the environment belong to the category of sensors for remote sensing.
Camera: used for recording images during the exploration of the environment. Generally it is featured in an Unmanned Aerial Vehicle (UAV) for photo shooting, placed in a special self-stabilized structure (the gimbal; Figure 2.4 shows an example[4]), in order to obtain a sharp image.
Lidar (Light Detection And Ranging): laser scanning useful for the navigation and the obstacle detection in autonomous vehicle. It measures the distance to a target by illuminating it with a ultraviolet, visible, or near

[1] NAVSTAR GPS, NAVigation Satellite Timing And Ranging Global Positioning System, `http://www.gps.gov/`

[2] `https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf`

[3] `https://www.cdiweb.com/datasheets/invensense/PS-MPU-6000A.pdf`

[4] `http://www.tarot-rc.com/index.php?main_page=product_info&cPath=65_96&products_id=1525`

Figure 2.4: Tarot - 2 axes Gimbal GoPro H3

Figure 2.5: LIDAR - Lite Laser Rangefinder [RB-Pli-01]

infrared laser light. The LIDAR-Lite Laser Rangefinder[5] is shown in Figure 2.5.

Radar (RAdio Detection And Ranging): like lidar sensors, it is used for autonomous robot navigation with obstacle avoidance. Radar is an object detection system which exploits radio waves to determine the range, angle, or velocity of targets. Figure 2.6 displays the Doppler radar sensor[6].

Environmental sensors

The sensors inherent to this category are capable of detecting the environmental conditions of the monitored area.

Temperature and humidity sensors: observe the temperature and humidity levels featuring the tracked area. A sensor for temperature and hu-

[5] `http://www.robotshop.com/media/files/pdf/rb-pli-01-datasheet.pdf`

[6] `https://www.parallax.com/sites/default/files/downloads/32213-X-BandMotionDetector-v1.1_0.pdf`

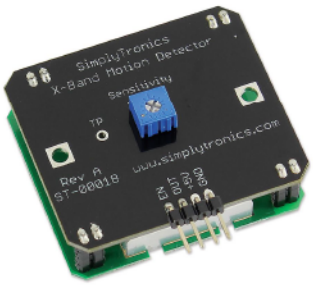


Figure 2.6: Doppler X band Radar detector sensor module

Figure 2.7: Grove - Temperature and Humidity Sensor

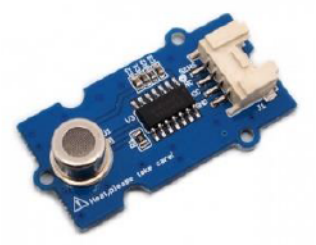


Figure 2.8: Grove - Air Quality Sensor

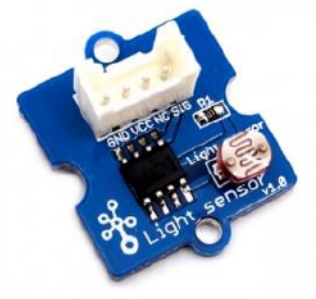


Figure 2.9: Grove - Light Sensor

Figure 2.10: Grove - Dust Sensor

midity detection[7] is presented in Figure 2.7: powered with voltages between 3.3 V and 6 V, it is able to detect moisture rates ranging from 5% to 99% and temperatures ranging between $-40°C$ and $+80°C$.
Air quality control sensor: measures the toxic gases concentration in the air. Figure 2.8 illustrates a sensor[8] capable of detecting smoke, volatile hydrocarbons and other gaseous chemical compounds.
Light sensor: outputs a voltage or current proportional to the light intensity in the environment. Figure 2.9 shows a photoresistor[9] powered with a voltage between 3 V and 5 V.
Dust sensor: is able to provide information on the air quality related to the amount of particulate. An example[10] is depicted in Figure 2.10.
Barometer sensor: detects the atmospheric pressure of the monitored environment. Commercially available components often integrate barometer and thermometer, as the example[11] shown in Figure 2.11, with I2C (Inter-Integrated Circuit) connection.
Digital airspeed sensor: used for the detection of the air speed around an aircraft, it is particularly useful in conditions of strong wind or slow flight, as it enables self-stabilization of the aircraft. Figure 2.12 displays an airspeed sensor[12] usable with different microcontrollers.

Tracking sensors

Technologies to automatically identify and track transponders (*a.k.a.* tags) attached to objects belong to this sensor category.
Radio-Frequency IDentification (RFID): uses electromagnetic fields to

[7] `http://www.seeedstudio.com/wiki/Grove_-_Temperature_and_Humidity_Sensor`
[8] `http://www.seeedstudio.com/wiki/Grove_-_Air_Quality_Sensor`
[9] `http://www.seeedstudio.com/wiki/Grove_-_Light_Sensor`
[10] `http://www.seeedstudio.com/wiki/Grove_-_Dust_Sensor`
[11] `http://www.seeedstudio.com/wiki/Grove_-_Barometer_Sensor`
[12] `https://www.nxp.com/files/sensors/doc/data_sheet/MPXV7002.pdf`

Figure 2.11: Grove - Barometer Sensor

Figure 2.12: Digital Airspeed Sensor

improve automatic environment recognition capabilities. An RFID system consists of two essential components: (i) tags associated with objects; (ii) the tag reading or writing data in its memory. Figure 2.13 shows an example of RFID system used to track clothes.

Near Field Communication (NFC): small subset of the many standardized RFID technologies. In particular, NFC belongs to the family of High Frequency technologies, operating at a frequency of 13.56 MHz. Figure 2.14 shows a NFC tag.

Figure 2.13: RFID system

Figure 2.14: NFC tag

2.2.2 Actuators

According to the monitored and detected context, a smart object must be able to dynamically perform a corresponding adaptation of its behaviour and act on the surrounding environment in order to suitably modify it by exploiting reaction and enforcement capabilities. This is achieved by the use of specific components called *actuators*. An actuator is a hardware device able to convert a controller command signal into a change in a physical parameter. The triggered change is usually mechanical (*i.e.*, position or velocity). An actuator is typically activated by a low-power command signal, so an amplifier may be required to provide sufficient power to drive the actuator. Various forms of actuators are available in order to meet particular requirements of process control. They can be classified into three main categories, presented below.

Electrical actuators

Examples of electrical actuators are switching devices such as diodes, transistors, triacs and relays. They are fed by a low-energy command signal generated by the controller; based on this input, they switch on or off or modulate associated electrical devices such as motors, valves, and heating elements.
Relay: used to control a circuit *e.g.*, for powering a remote lamp. The relay module[13] shown in Figure 2.15 is powered by 220 V and 10 A.
Electric motor: according to the received signal, drives another device, regulates speed, varies rotation direction or axis position. Figure 2.16 shows a direct current (DC) servomotor[14].

[13]Datasheet: `http://www.microbot.it/documents/mr009-001_datasheet.pdf`
[14]`https://inverterdrive.com/file/RS-Servo-Motor-Manual`

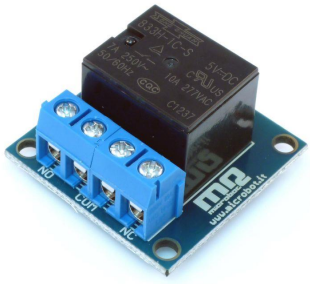


Figure 2.15: Relay module MR009-001.1

Figure 2.16: Parvex DC servo motor with encoder

Hydraulic actuators

An hydraulic actuator consists of a cylinder or fluid motor which uses hydraulic power to amplify the controller command signal.
Hydraulic cylinder: allows to transform the hydraulic energy, generated by a motor pump, into mechanical energy to achieve a linear force capable of moving a load. Hydraulic cylinders are employed in various applications such as weirs, dams, hydroelectric power stations, antiseismic systems, hydraulic cranes, aerial work platforms, compactors, drilling machines, tunnels, steering systems and many others. Figure 2.17 illustrates a hydraulic cylinder[15] with working pressure up to 210 bar.

Pneumatic actuators

A pneumatic actuator converts energy formed by compressed air at high pressure into either linear or rotary motion.
Rotary actuator: produces a rotary motion or torque. These actuators are generally installed on mechanical arms for industrial applications. In Figure 2.18 a rotary actuator[16] is presented.

[15] `http://ph.parker.com/it/it/hydraulic-cylinders-210-bar-2-h-series`

[16] `http://www.airwork.it/media/pdf/movement/attuatori-pneumatici-rotanti.pdf`

Figure 2.17: Parker hydraulic cylinder

Figure 2.18: Airwork's rotary actuator

2.2.3 Computing boards

A computing board is a complete computing platform built on a circuit board, endowed with microprocessor(s), storage, communication interfaces, input/output (I/O) channels and other features required for a standard functional calculator. Computing boards are characterized by several features that make them perfectly suitable for pervasive contexts:

- portability;
- low power consumption;
- cost effectiveness;
- versatility;
- compatibility.

The most relevant commercially available computing board solutions are analyzed hereafter.
Arduino: Arduino[17] is an open-source electronics platform employed in thousands of different projects and applications. It is a low-cost platform featured by limited computational resources. Figure 2.19 shows Arduino Due[18] microcontroller board. It is based on a 32-bit ARM Cortex M3 core

[17] `https://www.arduino.cc/`
[18] `https://www.arduino.cc/en/Main/ArduinoBoardDue`

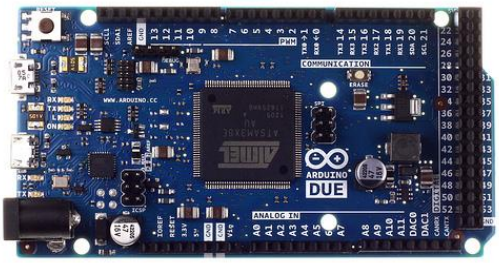


Figure 2.19: Arduino Due board

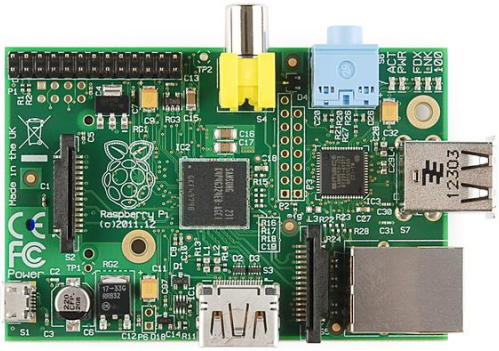


Figure 2.20: Raspberry Pi 2 Model B

microcontroller with 84 MHz clock frequency, 96 KB SRAM and 512 KB of Flash memory. It has 54 digital input/output pins (of which 12 can be used as Pulse-Width Modulation (PWM) outputs), 12 analog inputs, 4 UARTs (Universal Asynchronous Receiver-Transmitter hardware serial ports), an USB OTG capable connection, 2 DAC (digital to analog) outputs, 2 TWI (Two Wire Interface) pins for supporting the communication with TWI devices, a power jack, an SPI (Serial Peripheral Interface) header for support SPI communication using the SPI library, a JTAG (Joint Test Action Group) header for support the communication with devices implementing JTAG standards, a reset button and an erase button. There are several Arduino boards in different formats and configurations, and also there are components (called *shield*) providing sensors, actuators and communication interfaces that can be easily integrated into the Arduino board.

Raspberry Pi: Raspberry Pi[19] is a credit-card-sized single board computer developed by the UK based Raspberry Pi Foundation. It has a slightly higher cost but more superior computational resources with respect to Arduino

[19] `https://www.raspberrypi.org/about/`

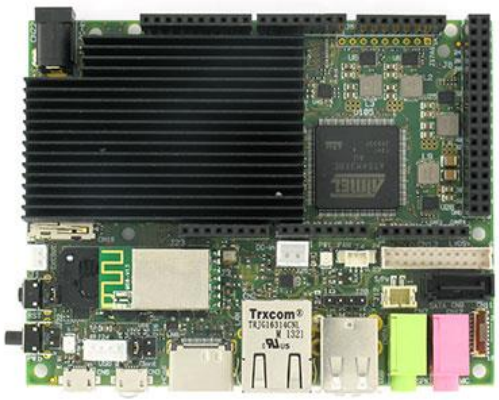


Figure 2.21: UDOO Quad

board and, for these reasons, it has become very popular among single-board computers for fast prototyping projects. Figure 2.20 illustrates Raspberry Pi 2 Model B[20]. It runs Linux on a 900 MHz quad-core ARM Cortex A7 processor, has 4 USB ports to connect peripherals HDMI and/or RCAvideo output, MicroSD card slot for mass storage and an Ethernet port.
UDOO: UDOO[21] is a family of open-source single board computers with the Android/Linux programming in order to create a functional all-in-one embedded system. Figure 2.21 presents the UDOO Quad[22] board. It is equipped with NXP i.MX 6 ARM Cortex A9 quad core CPU at 1 GHz clock frequency, 1GB DDR3 RAM, a 10/100/1000 Ethernet jack, a WiFi module, 2 Micro USB, 2 USB type A and one USB connector, a HDMI port, an on board MicroSD card reader and 76 fully available GPIO pins (62 digital and 14 digital/analog).

2.2.4 Robots

Smart objects can often be characterized as robotic agents driven by a computer program. In general, the word 'robot' refers to a programmable, multifunctional manipulator designed to perform a variety of tasks through various programmed motions [45].

Robotics has a strong impact in supporting a wide range of applications

[20] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[21] http://www.udoo.org/

[22] http://shop.udoo.org/eu/quad-dual/udoo-quad.html?___from_store\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{u\global\mathchardef\accent@spacefactor\spacefactor}\accent22u\egroup\spacefactor\accent@spacefactorsa&popup=no

Figure 2.22: TRP2-FOB - created by Finmeccanica Company

including manufacturing, exploration, military tasks, healthcare, entertainment, and many others. In all these application areas, robots are used to help or even replace human operators in repetitive and boring jobs (*e.g.*, in automated production processes such as assembly lines) or in more complex tasks requiring speed, strength and/or precision beyond physical human capabilities or hazardous environments (*e.g.*, in the presence of toxic substances, radioactivity, in the absence of oxygen, in underwater environments, in minefields, in inaccessible environment as a result of disasters, earthquakes, accidents or terrorist attacks, etc).

Many different forms of robots have been designed and developed such as drones, landers, rovers, atmospheric probes and robot arms. Specifically, three different types of robotic entities have been considered as natural smart object candidates within the vision proposed in this thesis: (i) unmanned ground vehicle (UGV - Figure 2.22 shows an example[23]), (ii) unmanned aerial vehicle (UAV - an example[24] is depicted inFigure 2.23) and (iii) unmanned underwater vehicle (UUV - Figure 2.24 displays an instance[25]). The specific form chosen during robot design is directly affected by the tasks it must perform and the characteristics of the environment in which it will operate. An important connotation whereby robots can be classified regards the level of autonomy they must prove [34]. Autonomy means the ability to work in dynamic and unstructured environments without requiring continuous human

[23] `http://www.leonardocompany.com/documents/63265270/66967378/body_TRP2_combat_2013_REV01.pdf`

[24] `http://www.ga-asi.com/Websites/gaasi/images/products/aircraft_systems/pdf/PredatorXP021915.pdf`

[25] `http://saab.com/naval/underwater-systems/autonomous-underwater-vehicles/double_eagle_sarov/`

Figure 2.23: Predator XP - created by General Atomics Aeronautical Systems

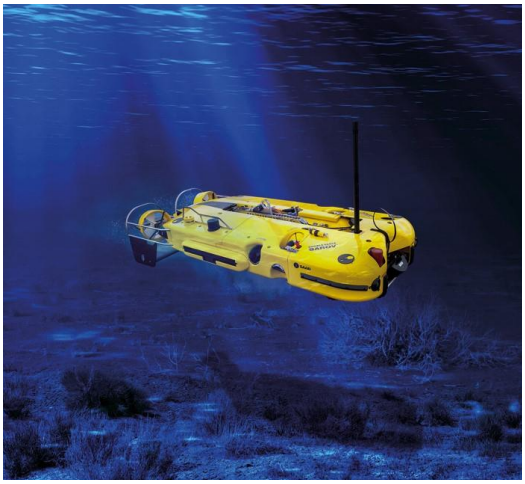


Figure 2.24: AUV/ROV HYBRID Double Eagle SAROV - created by Saab

intervention. In such environments many specific operating conditions are not known in advance, forcing the robot or any autonomous system to be able to detect the relevant features of the current situation and to trigger actions or make interventions accordingly.

Autonomy is defined as the ability of a robot to perceive, plan and act without external control [10]. In [28], the authors assess robot autonomy by taking into account four generic functions:

- **monitoring**: scanning through sensors;
- **development**: formulation of options or strategies to achieve specific aims;
- **selection**: decision about an option or strategy;

- **implementation**: execution of a chosen option.

Their classification establishes ten levels of autonomy, ranging from manual control to full automation. By increasing the degree of robot autonomy, its skills improve in terms of:

- exception handling, *i.e.*, the ability of reacting to unforeseen events minimizing the damages;
- management of ambiguous or contradictory perceptions;
- importance recognition with reference to various aspects of a situation;
- identification of similar aspects among different situations and, vice versa, different ones among similar situations.

In most of current robotic systems, the programmer must be able to foresee a variety of situations concerning the modes of machine reaction. Systems belonging to the highest-level of the classification are today difficult to achieve.

Robotic explored solutions

In this work, a review of the available literature was carried out proceeding according to a detailed analysis of the strengths and weakness of the various commercially available programmable robotic platforms. A summary of the main findings including a UGV, three UAVs and an autopilot for drone flight control, is reported in Table 2.1.

Table 2.1: Robotic explored solutions

Solutions	**Features**
iRobot Create2	– Size: 34 cm in diameter, 9.2 cm in height

	– Weight: 3.6 kg – Programmable Robot via Open Interface Commands – Serial-To-USB Cable – Rechargeable Battery – Self-Charging Home Base – Built-in sensors: Omnidirectional IR sensor, Left and Right Bumpers, 3 Wheel Drop sensors, 4 Cliff sensors, Wall sensor, 2 Wheel Encoders – Actuators: Left and Right Wheel Motors, Speaker, Bi-color Power LED, Play LED, Advance LED, Low-side Drivers on the BAM, Digital Outputs on the BAM
3D Robotics Iris^{+} drone 	– Motor to motor dimension: 550 mm – Height: 100 mm – Weight (with battery): 1282 g – Average flight time: 10-15 minutes – Payload capacity: 425 g – Battery: 3-cell 11.1 V 3.5 Ah lithium polymer with XT-60 type connector – Propellers: (2) 10 x 4.7 normal-rotation, (2) 10 x 4.7 reverse-rotation – Motors: AC 2830, 850 RPM/V – Radios available in 915 mHz or 433 mHz – 32-bit Pixhawk autopilot system with ARM Cortex M4 processor – uBlox GPS with integrated magnetometer
DJI Matrice 100 drone	– Weight (with TB47D battery): 2355 g

	– Diagonal Wheelbase: 650 mm – Max. Takeoff Weight: 3600 g – Operating Temperature: $-10°C$ to $40°C$ – Max. Speed: 22 m/s (no payload, no wind) – Hovering Time (with TB47D battery): No payload: 22 min; 500g payload: 17 min; 1 kg payload: 13 min – Hovering Time (with two TB47D batteries): No payload: 33 min – Battery: Dual Battery Compartments, Intelligent Flight Battery, 4500 mAh capacity, 22.2 V LiPo 6S – Fully programmable using SDK DJI – Supported Platform: Linux, Windows Embedded systems
3DR Pixhawk Autopilot 	– 32 bit ARM Cortex M4 Processor running NuttX RTOS – 14 PWM/servo outputs (8 with failsafe and manual override, 6 auxiliary, high-power compatible) – Connectivity for additional peripherals: UART, I2C, CAN – Integrated backup system for in-flight recovery and manual override with dedicated processor and stand-alone power supply – Backup system integrates mixing, providing consistent autopilot and manual override mixing modes – Redundant power supply inputs and automatic failover – External safety button for easy motor activation – Multicolor LED indicator – High-power, multi-tone piezo audio indicator – MicroSD card for long-time high-rate logging

2.3 Knowledge acquisition, representation and sharing

In order to annotate a context profile, a smart object has to monitor the data flow collected by onboard object sensors, as well as by Wireless Sensor Network (WSN) possibly present in the surroundings. The analysis of gathered data is carried out by means of data mining approaches. Knowledge Representation tools and technologies are borrowed from the Semantic Web

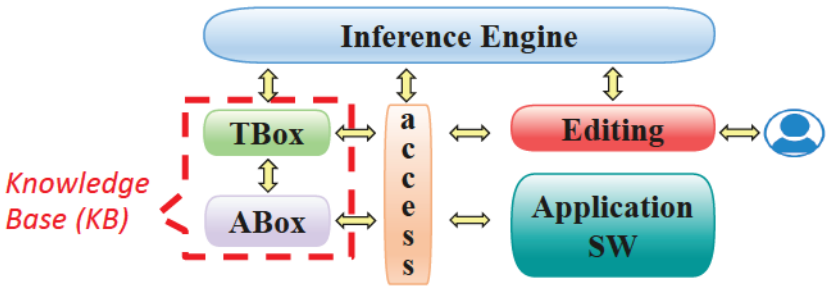


Figure 2.25: Knowledge Representation System

initiative and adapted to pervasive smart devices in order to integrate novel high-level knowledge-based capabilities.

2.3.1 Knowledge-based systems

Knowledge Representation (KR) [58] is a field of Artificial Intelligence (AI) formalizing knowledge about a fragment of reality by means of logic-based languages, making it machine-understandable, in order to enable the creation of software agents able to perform automated *reasoning* procedures.

Deductive reasoning (also known as inference) derives a final expression (*conclusion* or *thesis*) starting from a set of initial expressions (*premises* or *assumptions*) and assuming the truthfulness of the statements.

A Knowledge Representation System (KRS) in a particular domain provides the means to build Knowledge Base (KB), reason upon its content and manipulate it. As shown in Figure 2.25, a KRS consists of five main components:

- Knowledge Base (KB): repository to store complex structured and unstructured information. It is a pair $K = <\mathcal{T}, \mathcal{A}>$. $\mathcal{T}$ is the *Terminological Box* (TBox or *ontology*), *i.e.*, a formal representation of the conceptual model of a domain. On the other hand, $\mathcal{A}$ is the *Assertion Box* (ABox), specifying the factual information of a particular scenario within the domain as a set of *individuals*, whose descriptions use classes and properties from the ontology. In classical KR approaches, KBs are centralized and monolithic, stored in a given server location reachable by several clients interested in performing operations on it.

- Access Interface: shared boundary to allow the access to the knowledge contained in TBox and ABox by software applications.

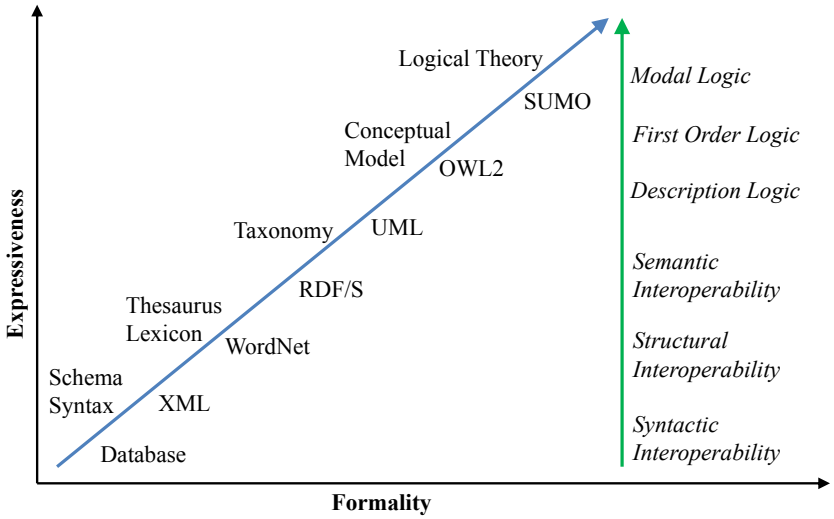


Figure 2.26: Knowledge Representation Languages

- Inference Engine: set of reasoning services provided by a software component called *reasoner*, allowing to deduce implicit knowledge from that expressed in the KB.

- Editing Interface: software application to provide the management of the contents in TBox and ABox by a human operator.

A key parameter for defining and creating a KR system is represented by its expressivity. The more expressive a KR results, the easier and more compact it is to express a fact or element within the semantics and syntax of that KR. However, more expressive languages usually require more complex logic and algorithms to construct equivalent inferences. A highly expressive KR is also less likely to be complete and consistent; whereas less expressive KRs may be both complete and consistent.

In literature, several KR languages and standards have been developed for defining ontologies on the Semantic Web, including Resource Description Framework (RDF), RDF Schema, Topic Maps, DARPA Agent Markup Language (DAML), Ontology Inference Layer (OIL), and Ontology Web Language (OWL) [53]. Figure 2.26 shows the relationship between KR languages in terms of formality and expressiveness. SUMO, an open-source declarative programming language, resides on the higher end of the scale. Formal languages such as DAML, OIL, and OWL are geared towards classification. It is important to emphasize that as the languages become more expressive, they require more complex software to support them. In the following subsections, Description Logics (DLs) family is presented in detail.

2.3.2 Description Logics

Description Logics (DLs) are among the most used KR languages. They are a family of formalisms arising from First Order Logic (FOL)[13, 26]. DLs allow to represent knowledge by means of:

- *concepts a.k.a. classes*, representing sets of objects;
- *properties a.k.a. roles*, representing relationships between pairs of concepts;
- *individuals*, *i.e.*, named instances of classes.

These elements can be combined in expressions via *constructors* to create DL expressions, specified in a formal semantics which associates an interpretation Ito each term. Concept *conjunction* is interpreted as set intersection: $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$. Concept *disjunction* is interpreted as set union: $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$. The connector $\neg$, if present, is the interpretation of the complement operator. More constructs exist which distinguish each language of the DL family and also determine both its expressivity and computational complexity characteristics.

An ontology is composed by two types of assertions: *inclusion*, which allows to define *is-a* relationships between classes; *equivalence*, which allows to give a name to a particular concept expression.

Description languages are distinguished by the constructors they provide. The *Attributive Language*($\mathcal{AL}$) has been introduced in [67] as a minimal language that is of practical interest. Constructs of $\mathcal{AL}$ are reported in what follows:

- $\top$, *universal concept*. All the objects in the domain.
- $\bot$, *bottom concept*. The empty set.
- A, *atomic concepts*. All the objects belonging to the set A.
- $\neg A$, *atomic negation*. All the objects not belonging to the set A.
- $C \sqcap D$, *intersection*. The objects belonging to both C and D.
- $\forall R.C$, *universal restriction*. All the objects participating in the R relation whose range are all the objects belonging to C.
- $\exists R$, *unqualified existential restriction*. There exists at least one object participating in the relation R.

In order to increase the expressiveness, the $\mathcal{AL}$ language can be extended with additional constructs such as:

- $\mathcal{N}$ $(\geq nR)$[26], $(\leq nR)$, $(= nR)$[27], *unqualified number restrictions*. Respectively the minimum, the maximum and the exact number of objects participating in the relation R.
- $\mathcal{U}$ $C \sqcup D$, *concept union*. The objects belonging to C or D.
- $\mathcal{E}$ $\exists R.C$, *full existential qualification*. Existential restrictions that have fillers other than $\top$.
- $\mathcal{C}$ *complex concept negation*. Negation of concepts that are are comprised of other concepts.
- $\mathcal{I}$ R^{-}, *inverse roles*. Refer to the inverse of a binary relation.

The formula $\mathcal{AL}[\mathcal{N}][\mathcal{U}][\mathcal{E}][\mathcal{C}][\mathcal{I}]$ is used to indicate the $\mathcal{AL}$ language extensions.

This work adopted the *Attributive Language with unqualified Number restrictions* ($\mathcal{ALN}$) DL. It provides adequate expressiveness while keeping polynomial complexity, both for standard and non-standard inferences. Table 2.2 summarizes syntax and semantics of constructors and assertions in $\mathcal{ALN}$.

Table 2.2: Syntax and semantics of $\mathcal{ALN}$

Name	**Syntax**	**Semantics**
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$
Universal quantification	$\forall R.C$	$\{d_1 \mid \forall d_2 : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow \ d_2 \in C^{\mathcal{I}}\}$
Number restrictions	$\geq nR$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \geq n\}$
	$\leq nR$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \leq n\}$
Inclusion	$A \sqsubseteq D$	$A^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Equivalence	$A \equiv D$	$A^{\mathcal{I}} = D^{\mathcal{I}}$

[26]Notice that $\exists R$ is equivalent to $(\geq 1R)$

[27]Notice that $(= nR)$ is a shortcut for $(\geq nR) \sqcap (\leq nR)$

Web Ontology Language (OWL)

Web Ontology Language (OWL) [1] is W3C Recommendation with the purpose of creating ontologies and expressing metadata for resources in the Semantic Web. OWL uses the Internationalized Resource Identifier (IRI) for resource nomenclature and is based on the Resource Description Framework (RDF) [68] in order to grant the following capabilities to ontologies:

- ability to be distributed across many systems;
- scalability to Web needs;
- compatibility with standard Web for accessibility and internationalization;
- openness and extensibility.

OWL branches into three different levels of complexity, from the least to the most expressive:

- *OWL Lite*, allows very basic taxonomy and constraints definition;
- *OWL DL*, enables a fairly ample expressiveness retaining computational decidability. The name indicates a direct correspondence between OWL and Description Logics (DLs);
- *OWL Full*, provides the highest level of flexibility and expressiveness, sacrificing computational decidability.

OWL Lite syntax is a subset of *OWL DL*, which is a subset of *OWL Full*. The subset of *OWL DL* tags allowing to express the $\mathcal{ALN}$DL is presented in Table 2.3. In pervasive scenarios featured by volatile nodes interacting in an opportunistic fashion in order to achieve a common goal, *OWL Full* is not suitable because of its undecidability, while suitable fragments of *OWL DL* provide a good trade-off between expressiveness and complexity. In this work the domain of interest was modeled using the latest specification of OWL language, *i.e.*, OWL 2 [1], announced on 27 October 2009. It supports a variety of syntaxes to read, store and exchange knowledge conceptualization among applications.

OWL 2 includes three sublanguages, named profiles: (i) *OWL 2 EL*, a fragment that has polynomial time reasoning complexity; (ii) *OWL 2 QL*, designed to enable easier access and query to data stored in databases and (iii) *OWL 2 RL*, a rule subset of OWL 2. Each profile applies for a specific use and offers a different tradeoff between expressiveness and reasoning efficiency. Unlike OWL sublanguages, OWL 2 profiles are independent among them.

Table 2.3: Correspondence between OWL and DL syntax

OWL syntax	**DL syntax**
owl : Thing	$\top$
owl : Nothing	$\bot$
owl : Classrdf : ID = "C"	C
owl : ObjectPropertyrdf : ID = "R"	R
rdfs : subClassOf	$\sqsubseteq$
owl : equivalentClass	$\equiv$
owl : disjointWith	$\neg$
owl : intersectionOf	$\sqcap$
owl : allValuesFrom	$\forall$
owl : someValuesFrom	$\exists$
owl : maxCardinality	$\leq$
owl : minCardinality	$\geq$
owl : cardinality	$=$

2.3.3 Inference services for knowledge discovery

Semantic matchmaking [29] is the process of finding the best match between different resources with respect to a given request, in a context where both the resources and the request are annotated referring to the same ontology. Given a specific request (D) and a resource (S) different match classes (categories) are defined, as reported in Table 2.4 [19] w.r.t. an ontology $\mathcal{T}$. The most desired match is obviously the exact one, but from the viewpoint of a requester full match is equally acceptable. However, potential and partial matches are the most common in real complex scenarios.

DL reasoners usually provide two standard inferences for semantic matchmaking on concept expressions:

- *Subsumption*: checks if S is more specific than D w.r.t. the ontology $\mathcal{T}$. In a formal way: $\mathcal{T} \models S \sqsubseteq D$.
- *Satisfiability*: verifies if the conjunction of S and D is satisfiable w.r.t. the ontology $\mathcal{T}$. In a formal way: $\mathcal{T} \not\models S \sqcap D \sqsubseteq \bot$.

In both cases the output is boolean, hence these services can only provide 'yes/no' answers without an explanation of the outcome, so that a request refinement is not facilitated. Furthermore, Subsumption and Satisfiability manage only full matches, which are quite rare in complex realistic scenarios. In order to take into account approximate matches, non-standard inference services Concept Abduction and Concept Contraction can be used [61]:

- *Concept Abduction*: if $\mathcal{T} \models S \sqsubseteq D$ is false then Concept Abduction computes a concept H (for *Hypothesis*) such that $\mathcal{T} \models S \sqcap H \sqsubseteq D$ is true. H (for *Hypothesis*) represents missing features in the resource S, able to completely satisfy D w.r.t. the information modeled in $\mathcal{T}$. In semantic matchmaking problems under the Open World Assumption, such missing information does not represent a constraint of absence, but just underspecified information, *e.g.*, because unknown or deemed irrelevant.

- *Concept Contraction*: if the conjunction $S \sqcap D$ is unsatisfiable w.r.t. $\mathcal{T}$, *i.e.*, S and D are not compatible with each other, Concept Abduction cannot be used and Concept Contraction should be adopted. It is able to determine what features G (for *Give up*) can be retracted from D to obtain an expression K (for *Keep*) such that $K \sqcap S$ is satisfiable in $\mathcal{T}$. If nothing can be kept in D during the contraction process, one gets the worst solution $\langle G, K \rangle = \langle D, \top \rangle$, that is give up everything of D. On the opposite, if $S \sqcap D$ is satisfiable in $\mathcal{T}$ nothing has to be given up and the solution is $\langle \top, D \rangle$, *i.e.*, give up nothing.

For both these inference services minimality criteria for solutions are needed, which induce a distance metric (*penalty*) useful to yield a graded similarity between request and provided resources which enables in turn the resource ranking. Concept Contraction and Concept Abduction can be considered as extensions to *Satisfiability* and *Subsumption* standard inference services, respectively.

Table 2.4: Match classes

Name	**Description**	**Semantics**
Exact match	S ia semantically equivalent to D. All the requirements expressed in D are in S and S does not expose any additional feature w.r.t. D	$\mathcal{T} \models D \equiv S$
Full match	S is more specific than D. All the requirements expressed in D are provided by S and S exposes further characteristics both not required by D and not in conflict with the ones in D	$\mathcal{T} \models S \sqsubseteq D$

Plug-In match	D is more specific than S. All the characteristics expressed in S are requested by D and D exposes also other requirements both not exposed by S and not in conflict with characteristics in S	$\mathcal{T} \models D \sqsubseteq S$
Potential match	D is compatible with S. Nothing in D is logically in conflict with anything in S and vice-versa	$\mathcal{T} \not\models S \sqcap D \sqsubseteq \bot$
Partial match	D is not compatible with S. At least one requirement in D is logically in conflict with some characteristic in S	$\mathcal{T} \models S \sqcap D \sqsubseteq \bot$

By means of Concept Contraction and Abduction it is possible to move from a partial match to a full one by exploiting a query refining process:

$$\textbf{partial} \rightarrow \textbf{potential} \rightarrow \textbf{full}$$

2.3.4 Machine learning techniques for pervasive scenarios

Environmental information includes important facts a smart object can learn about its context. They are built from perceived data captured through sensors equipment and describe the evolution of observed properties over a time span. Unfortunately, raw sensor data –not accompanied by descriptive metadata– are of difficult exploitation, as they are hard to be interpreted or integrated in case of resource dearth. Data analysis and mining techniques, using machine learning and/or reasoning algorithms, allow eliciting and characterizing a more meaningful context description starting from large environmental datasets.

Machine learning classification algorithms are grouped by similarity in terms of their function as follows: (i) logical/symbolic learning methods based on the generation of intelligible rules (*e.g.*, Decision Trees [50] and Rule-based classifiers [32]), (ii) perceptron-based techniques inspired by the structure and/or function of biological neural networks [56] (*e.g.*, Artificial Neural Networks [57] and Radial Basis Function (RBF) networks [38]), (iii) statistical learning algorithms characterized by an explicit underlying probability model (*e.g.*, Bayesian Networks [73]), (iv) instance-based learning approaches defined by producing hypotheses directly from the training instances (*e.g.*, k-Nearest Neighbour [22]) and (v) other discriminative models

for classification and regression analysis based on a separating hyperplane (*e.g.*, Support Vector Machine [82]).

Various machine learning techniques have different model accuracy, as well as storage and computational requirements, hence no one is universally suitable for smart objects purposes. Anyway, useful classification surveys can be found in literature, such as [44]: a summary of the main findings is reported in Table 2.5.

It is evident that smart objects need not only high accuracy, but also computational efficiency. Particularly, *incremental learning*, *e.g.*, in the incremental version of k-Nearest Neighbors (k-NN) algorithm [15], appears to be very useful for pervasive intelligent agents which should evolve their understanding about the surrounding environment as new data come and are processed.

Table 2.5: Comparison of classification techniques

Technique	**Strengths**	**Limits**
Decision Trees [50]	– Resilient to noise – Tolerant to missing values – Very easy to interpret – Effective with discrete/nominal features	– Improper with continuous features – Risk of overfitting
Artificial Neural Networks [57]	– Effective with multidimensional and continuous features – Works well even with feature multicollinearity or nonlinear relationships	– High computational cost – Not tolerant to missing values
Bayesian Networks [73]	– Works also with small datasets – Fast training – Robust w.r.t. missing values	– Large number of features greatly increases computational and storage costs
k-Nearest Neighbors [22]	– Generally accurate – Insensitive to outliers – Works well with both nominal and numerical features – Can be used as incremental learner	– Relatively large storage costs – Not tolerant to noise and missing values

Support Vector Machines [82]	– Good accuracy – Works well with multi-dimensional and continuous features	– Requires a large training set – High computational complexity

2.3.5 Middleware platforms for knowledge sharing

Information sharing among several heterogenous smart entities cooperating in the same complex distributed environment can be achieved by exploiting middleware software infrastructures. This software architecture plays the role of an abstraction layer hiding details about hardware devices, network configuration and other lower-level software services from an application. Furthermore, middleware solutions provide means to discover services/resources within the network.

Discovery facilities of traditional Service Oriented Architecture (SOA) middleware –*e.g.*, CORBA [21] and UDDI [47]– are based on syntactic match between service IDs or encoded service attributes. Analogously, the majority of current middleware platforms is topic-based, relying on a set of predefined subjects. They ground discovery just on trivial string matching of topics and cannot support dynamic evolution of the available service types without any structured information about service characteristics. Those static configurations reveal a non-negligible limit in the exploitation of service-oriented approaches in fully autonomic and advanced application scenarios. For instance, the topic-based publish/subscribe protocol for smart objects cooperation in [31] lacks service description and discovery based on formal models.

Semantic-enabled service and request specification allows more accurate and flexible characterization of requesters' needs and providers' capabilities. Moreover, they can improve effectiveness and adaptability of all phases of service life cycle [40]. Semantic-based service/resource discovery represents an important requirement in mobile and pervasive computing contexts, where the computational limits and resource volatility require decentralized paradigms and dynamic lightweight middleware. Different semantically enriched middleware platforms exist in literature [48, 46]. However, they take into account only full matches, which are quite rare in complex pervasive domains with convoluted descriptions.

Supporting approximate matches and service ranking metrics is particularly relevant for fine-grained discovery in semantic matchmaking approaches. That is why ubiquitous logic-based matchmakers implementing non-standard inference services [63] appear as enabling technologies in mobile and pervasive

middleware infrastructures.

2.3.6 Stream reasoning

Stream reasoning [75] is a recent approach aiming to provide abstractions, foundations, methods and tools needed to deal with many real-world applications operating on rapidly changing data. In such scenarios, data is in the form of a continuous information stream (flow), rather than a static set stored in a repository. Examples include data collected by sensor networks and involving supply chain, road traffic or computer network traffic monitoring, the management of cellular networks, the control of financial transactions, the continuous health monitoring and many more.

In the mentioned application scenarios, a separation between the acquisition and processing phases is clearly impossible, as the speed of data generation is greater than the time needed to store it and the economic storage cost of data exceeds the benefits due to analysis. Stream reasoning replaces the classic one-shot query, typical of relational databases, with continuous queries, which return new results as soon as new data arrives from the stream. Moreover, traditional Database Management Systems (DBMSs) and the classic data manipulation algorithms are not suitable for managing numerous and complex queries on continuous data streams. Therefore, in the late 1990s the first Data Stream Management System (DSMS) were born.

In literature there are several approaches to combine existing DSMSs with the Semantic Web paradigm. The basic idea is to abstract from fine-grained data streams into aggregated events [24] by merging the existing data models, access protocols, query languages for DSMS with the Semantic Web.

Since SPARQL language (recursive acronym for: SPARQL Protocol and RDF Query Language) [78] was recommended as the standard language for RDF querying by the W3C, extension proposals were published for data stream management including Streaming SPARQL [12] and Continuous SPARQL (C-SPARQL) [8]. They introduce a new data type, the RDF stream. These languages are designed to express queries that are recorded and continuously performed on both RDF repositories and RDF streams. For example, they allow the execution of queries like: "How many cars are continuously entering the city center?".

In pervasive contexts, high-level context descriptions produced by smart objects constitute a continuous information flow variable over time, rather than a static data set. Equipping a smart object with a DSMS is currently unrealistic in IoT applications, as the architectural and performance requirements of a pervasive computing platform are very different. Nonetheless, it is possible to introduce some stream reasoning techniques based on semantics.

The approach in [60] is a general framework to provide a compact representation of large concept flows and identify common information atoms, in order to support pattern analysis and identification of significant trends. Therefore, stream reasoning techniques can provide the means to harness the flow of semantically annotated updates inferred from low-level data, in order to enable adaptive context-aware behaviors in a range of applications.

2.4 Current issues and limitations

The research line investigated in this thesis concerns the integration of intelligence into micro-devices deployed in pervasive environments and wirelessly interconnected within unstructured settings. The main goal is to give objects the ability to: (i) automatically extract and process environmental information; (ii) semantically annotate and enrich mined information like in a log about the context they operate in; (iii) make the gained knowledge easily accessible by other network nodes through publishing mechanisms akin to a blog for people exchanging information on the World Wide Web. Data sources can be either on-board sensors or queried through short-range wireless communication protocols. In the latter case, some reference technologies include sensor-enabled RFID (Radio Frequency IDentification) tags [39] and Wireless Sensor and Actor Networks (WSANs) [3]. Raw sensor data must be analyzed in a resource-efficient way to enable interpretation and annotation with descriptive metadata about relevant conditions, events and features of the current context.

Current event recognition approaches are mainly based on threshold detectors or standard machine learning techniques. Most of these works combine information from heterogenous sources and derive a rough estimation of the monitored context state. In [49], a Bayesian approach was proposed in order to determine an estimation of the user's situation in the workplace by deriving higher-level information from labeled sensor data. In [7], a distributed event detection technique based on decision tree classifier was proposed for disaster management fulfilling the requirements posed by resource limitations of WSNs. Although probabilistic learning models are characterized by the ability to handle noisy, uncertain and incomplete sensor data, they present several limitations such as scalability, ad-hoc static models and data scarcity. With the aim of overcoming the low level of model accuracy due to the use of crisp threshold values, proposals based on fuzzy logic were defined [42] for event detection leveraging a reduced-size rule base. The integration between low-level data analysis and high-level context interpretation is required for triggering actions, making decisions or intervening on the environment.

Knowledge-based approaches have been developed and adopted for making this feasible. In [18] and [51], ontology-based models were defined for home and office activity recognition, respectively. Both enabled intelligent mining with a high level of automation and exploited semantic-based reasoning supporting only full matches, which produce true/false match outcomes without explanation. This represents a significant limit in pervasive scenarios where partial matches occur between heterogeneous information sources scattered in the environment. The semantic-based framework in [35] represents an improvement on knowledge discovery in Web of Things contexts, nevertheless data gathering and annotation were performed via simplistic threshold-based classification.

Sharing machine-understandable knowledge among ad-hoc network nodes is accomplished through the adoption of a method to establish common reference ontologies, enabling all involved entities to communicate using the same vocabulary. In recent years, the task of describing sensor features and retrieving data through ontology-based formalisms has been faced by the Semantic Web research community. *OntoSensor* [59] and *SSN-XG* [20] are among the most relevant and widely used ontologies in this field. They are general enough to cover different application domains and are compatible with the OGC SWE (Open Geospatial Consortium Sensor Web Enablement) standards at the sensor and observation levels [14]. Many projects, *e.g.*, SPITFIRE [54], adopted such ontologies combining semantic and networking technologies to build full frameworks. Unfortunately, these domain conceptualizations are too large and complex to be processed by a single node in pervasive computing contexts where smart objects pool their semantic-based knowledge for self-coordination and collaboration. Strategies for modularizing terminologies become necessary: the goal is to segment the overall model into a set of coherent modules about different subtopics to be used independently, while still containing information about relations to the other modules. Solutions available in literature are strongly influenced by the specific applications. Unlike classical ontology modularization [74], IoT smart applications require a dynamic, problem-oriented approach compatible with resource-constrained devices. A relevant framework enabling ontology decomposition and run-time rebuilding is presented in [65], although it grants limited flexibility to run-time ontology distribution. Furthermore, it only supports semantic matchmaking based on Subsumption, hindering inference services which evaluate approximate matches.

In addition to high accuracy, another basic requirement of intelligent objects concerns the computational efficiency for working on pervasive computing platforms featured by a continuous information flow coming from heterogeneous data sources. To overcome QoS-aware smart object orchestra-

tion issues in such scenarios, [33] defined two heuristic methods (top-down and bottom-up) based on the direction of the data flow during a service composition process. Furthermore, [30] addressed the problem of semantic data flow compression in limited resource spaces by developing a scalable middleware platform to publish semantically annotated data streams on the Web through HTTP. Unfortunately, the above proposals only allow elementary queries in SPARQL fragments on RDF annotations. More effective techniques like ontology-based *complex event processing* [17] leverage a shared domain conceptualization in order to define and specify complex events, conditions and actions running on an event processing engine.

Acknowledging all the issues and limitations of current solutions, the proposal submitted in this thesis aims to a more principled and general solution supporting distributed knowledge mining, representation, management, sharing and discovery in advanced pervasive applications where mobile computing devices have minimum processing capabilities, a small storage and low-bandwidth communication capabilities.

Chapter 3

Object (b)logging: paradigm, architecture, methods

This chapter describes in detail the paradigms, architectures and methods defined and developed for the object (b)logging vision. Each task of the proposed framework is analyzed in order to provide a clear picture of the entire approach designed for high-level information representation, knowledge discovery, allotment and sharing in distributed scenarios populated by smart objects.

3.1 Object (b)logging vision and motivation

The *Object (b)logging* paradigm fits in the pervasive computing field, and particularly in the Semantic Web of Things (SWoT) where high level machine-understandable knowledge representation is associated to real-world objects, locations and events [62]. In this vision, object (b)logging provides means to make an object capable of sensing the environment and detecting events, describing itself and what surrounds it in a fully automatic fashion, exposing its descriptions to the outside world as in a blog and acting appropriately in cooperation with other objects deployed in the environment. The key aspect of object (b)logging is sharing the information produced and learned by all entities involved in the observed area: the *blog*. In [52], a blog is defined as a web-based environment gathering user entries (*e.g.*, thoughts, passions, discussions, etc) in order to improve information dissemination and sharing for informal learning practice. Similarly, this research envisions the blog as the collection of all the information learned and produced by every involved actor, through which it is possible to self-organize for achieving a specific common goal. More formally, a blog represents the set of semantically an-

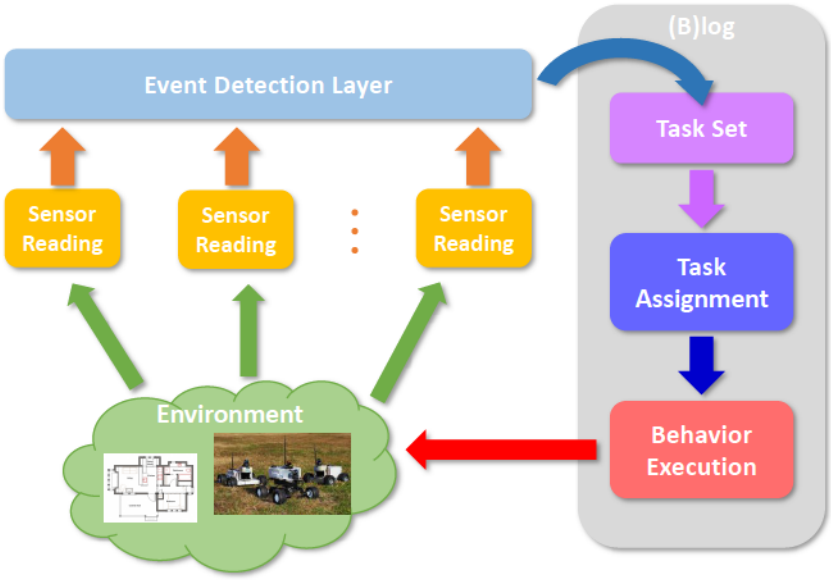


Figure 3.1: Architecture of the object (b)logging paradigm

notated descriptions w.r.t. a shared vocabulary and progressively enriched during object lifetime. It is based on a logic descriptive core and is used for intelligent interpretation of retrieved information.

Figure 3.1 shows the general object (b)logging architecture: each object immersed in a given environment collects data from sensors and processes them in order to produce a high-level annotation of detected events and conditions. By evaluating this descriptive information, implicit knowledge is derived for identifying the task set necessary to change the environment state. Then each object automatically infers which of its capabilities are useful in the examined scenario and acts on it accordingly, in a decentralized collaborative model. Mobile and pervasive scenarios are featured by severe resource limitations influencing processing, memory, storage and energy consumption. Hence, systems and applications should take into account hardware and software constraints on pervasive object capabilities.

The main contributions of the proposed approach can be summarized as follows:

1. Characterize the context of each smart object in a fully automatic way starting from a descriptive core and sensed environmental data, generating semantically rich and compact descriptions associated to contextual data. This is achieved by exploiting the integration of standard supervised machine learning techniques with non-standard semantic-based reasoning services.

2. Share the learned semantic-based knowledge within the network through the definition of a middleware architecture enriched with semantic layers for knowledge discovery, allotment and dissemination and based on the adoption of the ubiquitous Knowledge Base (u-KB) paradigm.

3. Achieve objects cooperation for triggering actions, taking decisions or making interventions on the environment by leveraging semantic-based matchmaking between available capabilities and required goals and tasks.

The proposed approach bridges the semantic gap between low-level observations and high-level detected phenomena enabling the development of pervasive knowledge-based systems with high degrees of autonomic capability not already allowed by conventional paradigms. Innovative analysis methods are defined for activity monitoring and recognition without requiring large computational resources and allowing to support flexible discovery strategies able to manage approximate matches in order to compensate for possible anomalies sensing and communication among spread entities.

The object (b)logging paradigm has a strong potential impact in supporting a wide range of applications. The main application areas are:

- *Urban search and rescue (USAR)*: intelligent autonomous robots are increasingly adopted in this field for facing operations which would be too hazardous for human agents in disaster scenarios (*e.g.*, search and rescue survivors trapped under collapsed buildings after earthquakes). Auto-coordination of different smart entities in a team allows to detect the context state, formulate plans to reach the mission goals and act accordingly.

- *Personal assistance*: orchestration of smart objects for the care of elderly and disabled people can provide assistance in their everyday life, improving daily functions at home or at work by automatically detecting the subject's current activity.

- *Industrial maintenance*: several entities self-coordinate to perform complex tasks for automated manufacturing, maintenance and inspection in industrial plants.

- *Home automation*: objects involved in this field include appliances, comfort security and surveillance management subsystems, even telepresence robots. By integrating intelligence into those objects, they become capable of scheduling themselves in an automatic fashion, according with the domestic environment and users' state.

- *Smart agriculture*: the application of wireless sensor networks and intelligent robots to agriculture allows to monitor the environmental parameters of fields helping farmers to improve crop quality and yield.

- *Entertainment*: smart objects can be deployed also for recreational or educational purposes learning to act appropriately as a result of a user's detected emotions.

For all these application fields, the proposed mining approach represents a paradigm shift with a tremendous potential impact in providing an advanced tool for analyzing raw environmental data gathered via heterogeneous sensing devices dipped in the environment, interpreting these information and associating an high-level characterization to the context, real-world objects, activities and events in resource-constrained pervasive scenarios. By leveraging the innovative aspects of the dissertation, the object (b)logging can increase effectiveness in sensing, interpreting, interacting and reacting to events and activities occurring in the physical world.

3.2 Framework and architecture

A rigorous stydy of available scientific literature in Semantic Web of Things has brought to design a comprehensive architecture for pervasive knowledge-based systems. In order to achieve objects self-description in complex mobile contexts, the proposed framework - illustrated in Figure 3.2 - consists of the following main tasks, carried out in a continuous process:

- processing and aggregation of raw data collected by sensors through data mining for perceiving the surrounding environment;

- association of semantically rich compact annotations to real-world retrieved data for high-level characterization of the information referring to an ontology providing the conceptualization for a particular domain;

- non-standard matchmaking services in order to obtain a logic-based representation of statistical data distributions so enabling a fine-grained event detection and advanced-level interpretation of knowledge to infer actions and interventions;

- advanced management of annotated data as a continuous stream of information through the support of stream reasoning techniques providing the means to harness the flow of semantically annotated updates progressively improved during the object's lifetime;

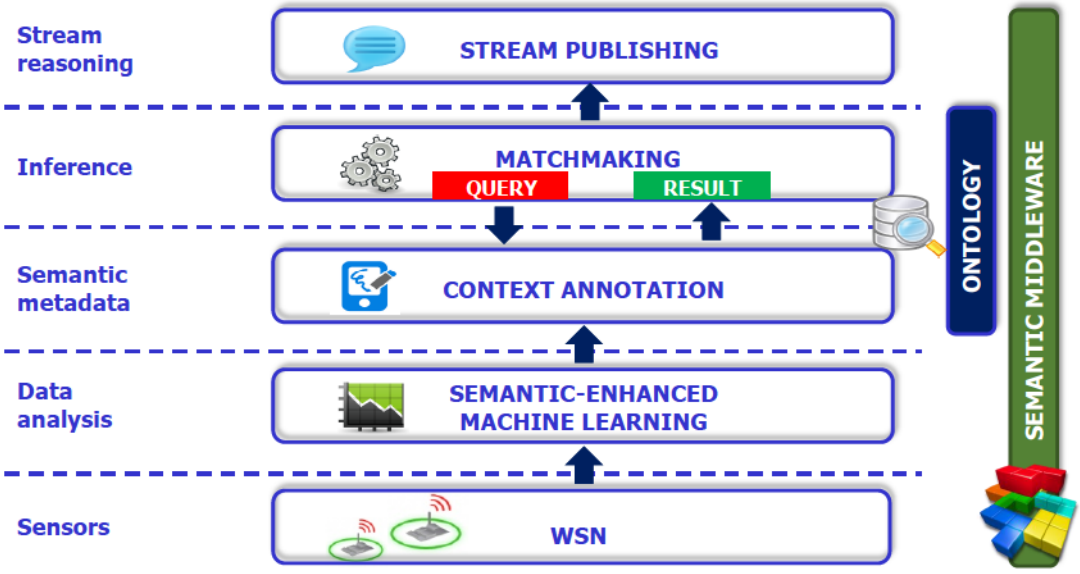


Figure 3.2: Block diagram of the proposed approach for object (b)logging

- inter-entity communication through lightweight middleware providing services for real-time decentralized and ubiquitous knowledge and service discovery.

As visible in the figure, there are two components cross the whole model: (i) ontology, shared between semantic annotation and matchmaking levels, and (ii) semantic middleware. In the following sections, each element of the proposal is discussed in detail.

3.3 Semantic-enhanced machine learning for context annotation

The envisioned framework aims to characterize the context of a smart object in a fully automatic way starting from raw sensor observations. Throughout the object's lifetime, this semantic endowment is progressively enriched, completed and exposed to the outside world as in a blog. Basically, a smart object classifies sensory data and further combines classifications to identify patterns, situations, activities and events featuring the real world. This high-level knowledge is useful in order to take decisions, trigger actions or make interventions on the environment where entities collaborate towards a common goal.

Raw data are annotated in a semantically rich formalism grounded on the $\mathcal{ALN}$DL [6]. Ontologies provide the needed conceptualization for each particular domain.Gathered data are classified using a knowledge-based evolution of k-Nearest Neighbor (k-NN) algorithm including non-standard inference support, as clarified hereafter. The algorithm is based on a distance metric of the current event featured w.r.t. a training set of data points, combining two components, named geometric score and contextual score. *Geometric features* describe the statistical distribution of the data while *contextual features* define the environmental characteristics which may influence the variation of observations. In order to compute both measures, each smart object continuously performs three steps, shown in Figure 3.3 and described afterwards:

1. **Clustering**. Unsupervised clustering is adopted to pre-process input data: an unknown input instance is associated to the nearest cluster description [80]. By applying *k-Means* algorithm [37], the system cleans data from noise and outliers, replaces missing values, identifies possible issues in clustering and makes a preliminary coarse data classification. Each cluster description includes two components: *geometry* and *context*. Geometry describes data through statistical parameters. The context component annotates data w.r.t. the adopted reference ontology.

2. **Advanced k Nearest Neighbors**. An enhanced version of the *k-NN* algorithm was devised to provide high-level data representation. It is based on a utility function combining semantic-based similarity measures with partial scores deriving from quantitative statistical attributes.

3. **Semantic-based matchmaking**. Matchmaking is performed on the data produced after the above steps in a given time span (observation window). The semantic annotation of the context is compared with descriptions of instances in the object Knowledge Base (KB) via inferences in [70] in order to find the most relevant environmental characterizations so giving a semantic-based interpretation to the raw data collected by sensing devices.

Each new data point or batch acquired in the observation window undergoes this process which terminates when the latest data points are integrated in the training set, while data older than a purging threshold are removed. This process allows to overcome the main issue with conventional classification techniques, *i.e.*, yielding only opaque labels without explicit and machine-interpetable meaning.

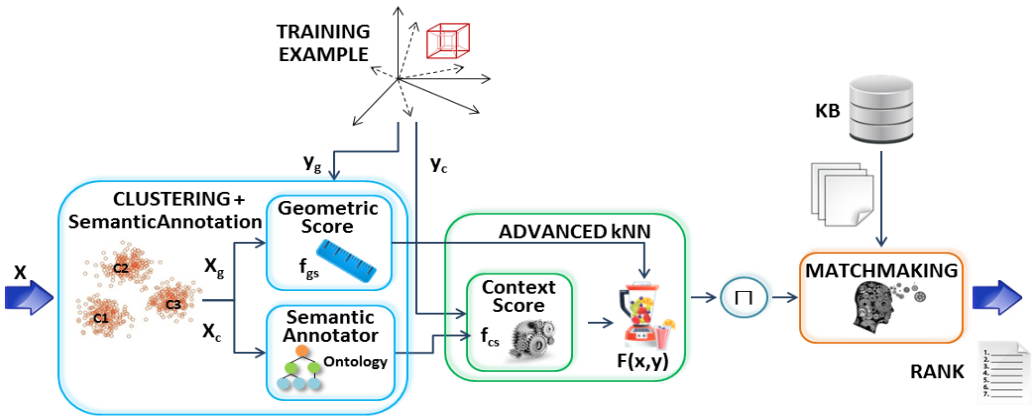


Figure 3.3: Block diagram of semantic-enhanced machine learning

Algorithmic details are explained in what follows.

3.3.1 Mining Approach

The integration of a classic k-NN supervised machine learning technique with semantic-based matchmaking is one of the notable aspects of the proposed approach. In standard k-NN [22], a data point is classified by assigning the most frequent label among the k training samples nearest to it, and a common metric is used for measuring the distance between the data point and each training instance. Figure 3.4 illustrates the k-NN algorithm: the input data point is represented by a red circle, while training instances are divided in two classes, depicted as blue and green circles, respectively. $F(x, y)$ is the common distance function exploited in the example presented in the picture. If the k parameter is set as $k = 3$, the most frequest label among the three training samples nearest to the input is green (2 out of 3 samples), so the input is classified as green. While, if $k = 7$, the most frequest label among the seven training samples nearest to the input is blue (4 samples out of 7), so in this case the input is classified as blue. In the thesis, the adopted distance metric integrates a geometric measure f_{gs} and a contextual semantic-based one f_{cs}, merged through a score combination function F. Here in what follows, input arguments of F are purposely named x and y, *i.e.*, the instance to be analyzed and each element of the training set, respectively. Both are described by geometric (x_g, y_g) and contextual (x_c, y_c) components.

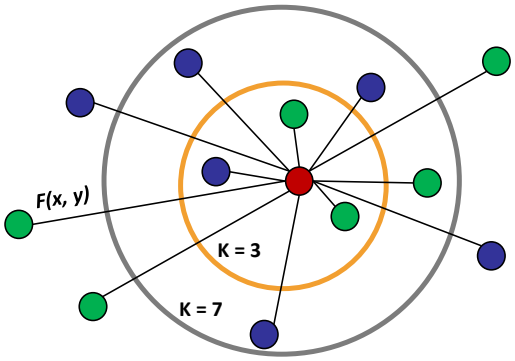


Figure 3.4: Illustrative example of the k-Nearest Neighbor (k-NN) algorithm

Geometric Score

As inspired by [55], the geometric score $f_{gs}(x_g, y_g)$ is a numerical assessment of how much y_g is similar to x_g. Similarity is referred to the statistical distribution parameters describing the raw data gathered for the current sample and for the training samples, *e.g.*, mean, standard deviation, *etc.*. Since x_g is the value to be matched, out of a set of n dimensions only the ones featuring x are considered. Therefore a basis vector $B(x_g) = \langle b_1, b_2, \ldots, b_n \rangle$ is defined, where $b_i \in [0, 1]$ and $b_i = 0 \Leftrightarrow x_{g_i} = \emptyset$. The matching value on a single dimension is:

$$dmatch(x_{g_i}, y_{g_{ji}}) = \begin{cases} \frac{|x_{g_i} \cap y_{g_{ji}}|}{|x_{g_i}|} & if\ B(x_{g_i}) = 1 \wedge \\ & B(y_{g_{ji}}) = 1 \\ 0 & else \end{cases} \tag{3.1}$$

The above value $dmatch(x_{g_i}, y_{g_{ji}})$ is determined by calculating the overlap between the two dimension values x_{g_i} and $y_{g_{ji}}$, *i.e.*, the i-th dimension of the j-th training example, divided by the size of x_{g_i}. This formula summarizes the notion that if $y_{g_{ji}}$ is fully contained in x_{g_i}, then $dmatch(x_{g_i}, y_{g_{ji}}) = 1$. If $B(x_{g_i}) = 0$ or $B(y_{g_{ji}}) = 0$, of course $dmatch(x_{g_i}, y_{g_{ji}}) = 0$. The overall geometric score is computed as:

$$f_{gs}(x_g, y_g) = 1 - \frac{\sum_{i=1}^{n} dmatch(x_{g_i}, y_{g_{ji}})}{n} \tag{3.2}$$

Note that dividing by n normalizes w.r.t. the maximum cardinality of x_g dimensions.

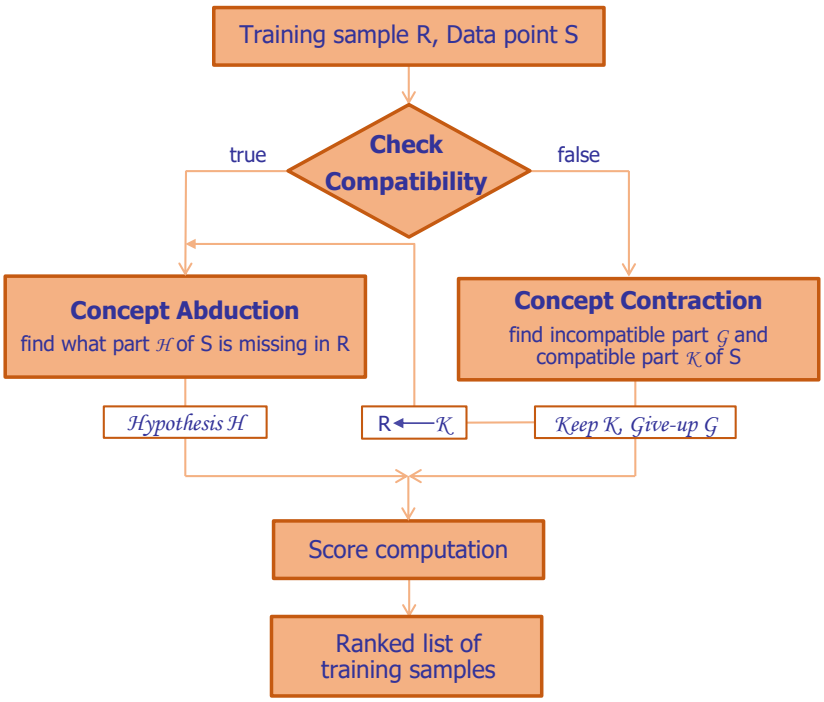


Figure 3.5: Semantic matchmaking algorithm

Contextual Score

The contextual metric $f_{cs}(x_c, y_c)$ is computed on features annotated in *OWL 2* language [77] according to the reference terminology. Score calculation is based on semantic matchmaking leveraging $\mathcal{ALN}$-based non-standard inference services.

The matchmaking process is summarized in Figure 3.5. It works on a semantically annotated description S of the measurement context for the current data point and the one for a training example R, both expressed w.r.t. an ontology. If they are compatible, *i.e.*, not in contradiction, Concept Abduction finds what part of S is missing in R and computes a concept *Hypothesis* that contains these features in order to completely satisfy it. Concept Abduction returns a value $penalty^{(a)}$ representing the associated semantic distance. Otherwise, if expressions are incompatible, Concept Contraction defines what characteristics can be retracted from the annotation of the training sample a concept *Give up* in order to obtain the remaining part of S, named *Keep*, which is compatible with R and computes a related $penalty^{(c)}$ value. The process goes on calculating the Concept Abduction between S and *Keep*. The reader is referred to Section 2.3.3 for algorithms and wider details argumentation about Concept Contraction and Concept

Abduction.

The contextual score is defined as:

$$f_{cs}(x_c, y_c) = \frac{\omega \cdot penalty^{(c)} + (1 - \omega) \cdot penalty^{(a)}}{penalty^{(a)}_{max}} \tag{3.3}$$

This value is normalized w.r.t. $penalty^{(a)}_{max}$, which is the maximum semantic distance from the data point: it is computed by comparing the annotation with the most generic concept $\top$ and it depends only on axioms in the reference domain ontology. The scoring mechanism is regulated by ω, which determines the relative weight of explicitly conflicting elements. In the proposed framework, ω depends on the calculated geometric score and is computed as:

$$\omega = \delta \cdot f_{gs}(x_g, y_g) \tag{3.4}$$

where $\delta \in [0.8, 1]$ is a proportional factor. The rationale is to assign assigns greater weight to Contraction penalty when the geometric distance is larger, since very different statistical properties of two samples require few or no explicit incompatibilities to be considered a close match.

Overall Score

The overall distance F is defined as:

$$F(x, y) = (f_{gs}(x_g, y_g) + \epsilon)^{\alpha} \cdot (f_{cs}(x_c, y_c) + \gamma)^{1-\alpha} \tag{3.5}$$

It is a monotonic function ranging between 0 and 1 providing a consistent ranking of input training examples. Lower outcomes represent better results. A tuning phase can be performed to determine the values of parameters α, γ, ϵ following requirements of a specific discovery application. More specifically, $\alpha \in [0, 1]$ weighs the relevance of geometric over contextual factors, respectively. Parameters $\epsilon \in [0, 1]$ and $\gamma \in [0, 1]$ control the outcome in case of either context or geometric full match. Geometric full matches occur when the statistical data input distribution is equal to the one of considered training example, while contextual full matches arise when all descriptive features of the data point measurement x_c are satisfied by the training example y_c.

3.4 Semantic-enhanced data stream analysis

In addition to the context annotation method described in the previous section, producing one annotation at a time, data stream analysis was devised

to summarize a flow of annotations into an evolving refined context characterization. The proposed data stream analysis approach is based on the development of a general-purpose framework to provide a compact representation of large flows of concepts allowing to identify common information components in order to support pattern analysis and the identification of significant trends [60].

The typical workflow of data mining and machine learning is preserved by the proposed approach: data collection and cleansing, model training, system predictions exploitation and evaluation. Nevertheless, semantic-based enhancements change the way each step is performed.

The starting point is represented by raw data collected by object onboard sensors or sensors dipped in a given environment and queried through short-range wireless communication protocols. These data are gathered for m different measuring parameters, generally named *features*. In order to support semantic-based data annotation and interpretation, an ontology $\mathcal{T}$ models the domain conceptualization along properly defined patterns. $\mathcal{T}$ is supposed acyclic and expressed in $\mathcal{ALN}$. This is required by the further adoption of non-standard inference services for semantic matchmaking. For each measuring parameter $\mathcal{T}$ will include a hierarchy of concepts (each one with its own properties), forming a partonomy of the top-most concept. In other words, each parameter will be represented via a classes/subclasses taxonomy featuring all significant configurations it can assume in the domain of interest. The depth of the hierarchy and the breadth of each level will be proportional to both resolution and range of sensing/capturing equipment, as well as to the needed degree of detail in data representation.

The proposal consists of three basic steps:

- **Training**: semantic-based representation of each possible output class that describes the observed phenomenon/event. Starting from these descriptions, a *training matrix* is built in order to track the occurrences of each atomic joint describing the event and extract a meaningful description.

- **Classification**: estimate of the distance computed between the semantic instance of the semantically annotated input to be classified and the event descriptions formulated during the training phase.

- **Evaluation**: system validation to assess the quality of those produced annotations. This phase exploits standard statistical metrics for system performance evaluation, such as a confusion matrix.

In what follows, proposed framework details are provided for each phase.

Training

The goal of the training step is to build a semantic annotation for each possible output class, connoting the observed event/phenomenon according to input data. The logical concepts modeled in the ontology are joined in a conjunctive expression in order to obtain the event description. The training phase works on a set S of n training samples, each characterized by m features. The samples can be grouped from the streaming data by defining proper time windows. Assume that w distinct outputs exist in the training set. Each feature value is mapped to the most specific corresponding concept in the reference ontology $\mathcal{T}$. Therefore the i-th sample $\forall\ i = 1, \ldots, n$ is composed by: (a) m concept components $C_{i,1}, \ldots, C_{i,m}$ annotating its features; (b) an observed output O_i labeled with a class of the ontology. Samples are processed sequentially by Algorithm 1 in order to build the so-called *Training Matrix* $\mathcal{M}$ (the pseudocode uses a MATLAB-like notation for matrix access). $\mathcal{M}$ is a $(w+1) \times (k+1)$ matrix having all the different outputs on the first column, all the distinct concept components on the first row and, in each element, the number of occurrences of the column header concept component in the samples having the row header output.

$\mathcal{M}$ gives a complete picture of the training set. Each output class can now be defined as the conjunction of the concepts having greater-than-zero occurrences in the corresponding row. By doing so, however, even very rare concepts are included, which may be not significant in defining the class. Therefore it is useful to define a *significance threshold* T_s as the minimum number of samples where a particular concept must appear to be considered significant for the occurrence of a particular output. The structure of $\mathcal{M}$ suggests the possibility to define different thresholds for each output and for each feature, as

$$T_{s(i,j)} = \theta_{(i,j)} \, |S|$$

with $0 < \theta_{(i,j)} \leq 1 \quad \forall i, j$ being adaptive ratios computed through a cross-validation process on the input dataset. Customized thresholds allow to focus sensitivity on the features with highest variance and/or the outputs most difficult to predict.

This training approach produces a Knowledge Base with conceptual knowledge (TBox) modeled by human experts and factual knowledge (ABox) created automatically from the available data stream, with instances representing the events of interest the system should be able to recognize.

Algorithm: $M = generateTrainingMatrix\left(\langle \mathcal{L}, \mathcal{T}, S \rangle\right)$

Require:
– $\mathcal{L}$ Description Logic;
– acyclic TBox $\mathcal{T}$;
– training set $S = \{S_1, S_2, \ldots S_n\}$, with $S_i = (C_{i,1}, \ldots, C_{i,m}, O_i)\ \forall\ i = 1, \ldots, n$, where all $C_{i,j}$ and O_i are expressed in $\mathcal{L}$ and satisfiable in $\mathcal{T}$.
Ensure:
– $\mathcal{M} : (w+1) \times (k+1)$ matrix of occurrences of the concepts for each observed output, where k is the total number of distinct concepts appearing in S

1: $\mathcal{M} := 0$ // (1×1) matrix
2: $r := 1, c := 1$
3: **for** $i := 1$ to $|S|$ **do**
4: $u_r := \mathit{findConceptIndex}(O_i, \mathcal{M}(:, 1))$
5: **if** $u_r = \mathit{null}$ **then**
6: append a row to $\mathcal{M}$
7: $r := r + 1$
8: $u_r := r$
9: $\mathcal{M}(u_r, 1) := O_i$
10: initialize $\mathcal{M}(u_r, 2 : c)$ to zeros
11: **end if**
12: **for** $j := 1$ to m **do**
13: $u_c := \mathit{findConceptIndex}(C_{i,j}, \mathcal{M}(1, :))$
14: **if** $u_c = \mathit{null}$ **then**
15: append a column to $\mathcal{M}$
16: $c := c + 1$
17: $u_c := c$
18: $\mathcal{M}(1, u_c) := C_{i,j}$
19: initialize $\mathcal{M}(2 : r, u_c)$ to zeros
20: **end if**
21: $\mathcal{M}(u_r, u_c) = \mathcal{M}(u_r, u_c) + 1$ // update occurrences
22: **end for**
23: **end for**
24: **return** $\mathcal{M}$

Algorithm 1: Creation of the Training Matrix

Classification

The subsequent **classification** task exploits a semantic matchmaking process based on *Concept Contraction* and *Concept Abduction* non-standard inference services [61] in order to manage partial matches between request and resources.

In the proposed approach, data of the instance to be classified are first labeled w.r.t. the reference ontology as in the first step of training, then their conjunction is taken as annotation of the instance itself. Penalty values obtained from matchmaking are used to compute the *semantic distance* between the input instance and the event descriptions generated during training. The predicted/recognized event will be the one with the lowest distance. Semantic matchmaking produces ranked similarity measures, associated with a logic-based explanation. Therefore the prediction outcome has a formally grounded and understandable confidence value. This is a clear benefit w.r.t. many standard ML techniques which produce opaque predictions. Furthermore, the approach does not take the instance annotation directly as the output, because the inherent data volatility could lead to inconsistent assertions, which would be impossible to reason on.

Evaluation

System **evaluation** works with a validation set, consisting of several classified instances represented w.r.t. the same ontology used for building the training set. The goal is to check how much the predicted event class corresponds to the actual event associated to each instance of the validation set. Beyond classical tools such as confusion matrix and statistical performance metrics such as accuracy, precision and recall, the graded nature of the predictions can be exploited to evaluate, *e.g.*, the average semantic distance of the predicted class from the actual one, analogously to error measure in regression analysis.

If computing resources permit it, incoming test data can also be used to update the training matrix on-the-fly, in order to allow the model to evolve as new data is observed. A proper extension of the baseline Algorithm 1 requires a fading mechanism to enable the system to 'forget' the oldest training samples.

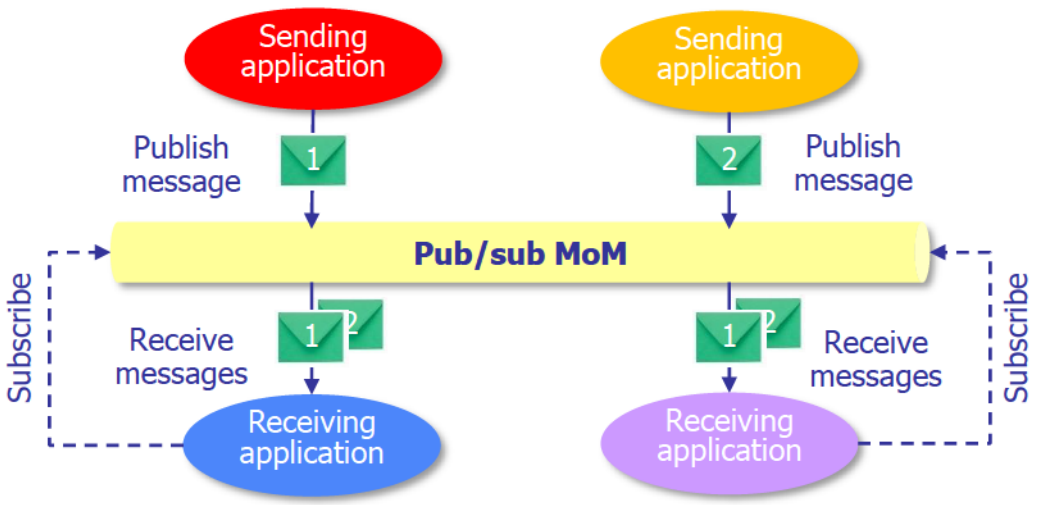


Figure 3.6: Pub/sub MOM middleware infrastructure

3.5 Semantic information sharing in pervasive scenarios

A key aspect of object (b)logging lies in the exploitation of an annotated blog for machine-understandable knowledge sharing. To achieve this, the proposed framework is materialized on top of the application-level of a publish/subscribe Message-Oriented Middleware (*pub/sub MOM*) infrastructure. This kind of middleware is fit for interconnecting large numbers of loosely coupled components on independent subsystems and devices (from now on *nodes*). In pub/sub MOM, interactions are based on the exchange of *messages* of arbitrary nature. Each message is labeled with a *topic*, *i.e.*, a string denoting the type, structure and/or purpose of the message payload. Each node can act as a *publisher* to emit messages with a specific topic and/or as a *subscriber* to receive all messages related to a subscribed topic. Typical middleware solutions guarantee message delivery with pre-defined levels of robustness, latency and security. Figure 3.6 depicts the model infrastructure of a pub/sub MOM. In conventional pub/sub MOM architectures, resource publishing and discovery occur through syntactic match of topics, which are opaque strings lacking any rigorous semantics. This is a great limitation in advanced applications, since it requires a previous knowledge of the network, the available resources and topics. Conversely, the proposed framework supports a dynamic semantic-based resource retrieval infrastructure through the integration of additional functional layers to an off-the-shelf MOM. As shown in Figure 3.7, the proposed approach includes three lay-

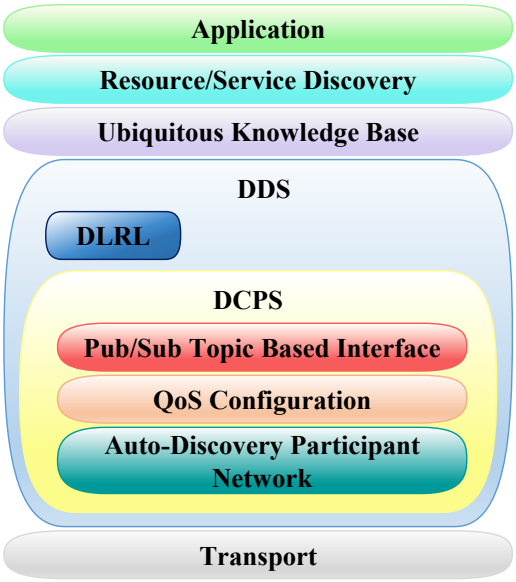


Figure 3.7: DDS Layered Architecture

ers: (i) Data Distribution Service (DDS), a standard protocol for pub/sub MOM; (ii) ubiquitous Knowledge Base, a distributed model for knowledge partitioning and on-the-fly materialization; (iii) Resource/Service Discovery, a decentralized collaborative resource/service discovery protocol exploiting non-standard inference services to enable a fine-grained categorization and ranking of resources matching a request. DDS provides services for real-time data distribution by adopting the publish/subscribe model in order to guarantee the basic inter-node communication. Its software infrastructure comprises two levels of interfaces:

- Data Centric Publish/Subscribe (DCPS): the lower layer of DDS defines entities, roles, interfaces and QoS policies for the publish/subscribe platform, as well as discovery techniques of communicating parties. Essentially, DCPS is the component related to the network communication.

- Data Local Reconstruction Layer (DLRL): optional upper level used for the integration of DDS into the application layer. DLRL allows to map exchanged data with application-layer object based on topics, in order to propagate updates automatically and transparently from the application to the network layer and vice versa.

On the top of DDS, two semantic layers are defined, developed and integrated in order to support decentralized and ubiquitous knowledge discovery. The

most significant details are reported in the following subsections.

3.5.1 Ubiquitous knowledge dissemination

Ttransparent access to information embedded in semantic-enabled devices of the network is granted by the u-KB layer.

As described in Section 2.3.1, static and centralized KBs are not realistic for IoT applications featured by volatile nodes interacting in an opportunistic fashion. The KB settings should comply with the intrinsic nature of the application scenarios and evolve toward a ubiquitous distribution of both terminology and factual knowledge. In other words, KBs should be partitioned in a decentralized way and scattered on multiple nodes. Specifically, in the proposed approach $\mathcal{T}$ is fragmented in one or more *chunks* managed by multiple distributed nodes, while individuals in $\mathcal{A}$ are not centrally stored, but disseminated in the environment as they belong to the endowment of each node. That is, a node is in charge of building and maintaining a logic-based description of its capabilities and features.

Due to the generality of the proposed approach, all nodes within the same network can manage any domain ontology, even using multiple vocabularies in order to cover different application domains. Furthermore, the use of unique ontology Uniform Resource Identifiers (URI) ensures that all objects working with the same reference ontology can share parts of the u-KB dynamically without requiring preliminary agreement among them.

As ontologies can be large and OWL adopts the verbose XML syntax, the u-KB approach exploits both compressed encoding and ontology partitioning. Compression and decompression are hidden to nodes, being performed automatically for all semantic message exchanged by the middleware extension. Any encoding algorithm or tool can be exploited, including standard ones like *gzip*[1] and EXI[2] as well as ontology-targeted ones [69].

Although compression of semantic annotations allows reducing information volumes noticeably, the intrinsic KB structure poses a significant data management issue in resource-constrained environments. A TBox –even when compressed– can still have a relevant size, particularly in articulated and complex domains. Nevertheless, in most cases it is practically unnecessary to materialize the whole ontology before reasoning, as used concept expressions refer just to a subset of the concepts and properties in the TBox. Terminology axioms can be split in a semantic-aware way, grouping them

[1] `http://www.gzip.org/`

[2] Efficient XML Interchange (EXI) Format 1.0 (Second Edition), W3C Recommendation 11 February 2014, `https://www.w3.org/TR/2014/REC-exi-20140211/`

in clusters with low inter-cluster correlation. In this way a node is able to retrieve only the TBox subset actually needed to perform a given reasoning task.

Due to the above considerations, a novel method for ontology partitioning is introduced here. It is based on associating each class with a unique ID, computed from its position in the taxonomy. The most generic class, always named *Thing* in OWL 2 (*a.k.a. Top* or $\top$ in DL notation), takes ID 1. Each nesting level adds a further numerical suffix, separated by a . (dot). An example of the proposed scheme is depicted in Figure 3.8. Subclasses of a given class have IDs differing just in the last number: in particular, sibling classes are sorted unambiguously in lexicographic order. This does not prevent ontology updates: as different versions of the same ontology are identified by different URIs according to Semantic Web best practices, conflicts are prevented. Such numbering system can be embedded directly in OWL ontologies via *annotation properties* associated to each class. In accordance with hierarchical ontology modeling patterns [72, Ch. 2], an *Upper Ontology* (UO) chunk is extracted from a given ontology, by taking the N topmost levels in the class hierarchy, which specify the most general concepts of a domain. N is a configurable parameter: its value should be chosen based on the ontology size, the desired chunk size and the depth of class nesting. Other chunks will consist of separate subtrees stemming from classes at level $N + 1$, so as to minimize overlap. In the example in Figure 3.8, if $N := 3$, then $Thing$, $Action$, $Entity$, $Environment$, $Event$, $FeatureOfInterest$, $Object$, $Quality$ will be included in the *UO Chunk*; the subtrees stemming from $FeatureOfInterest$ and $Object$ will take their own *Chunk A* (or more than one chunk if it is larger than a configurable threshold); the same occurs for the subtrees with other level-3 classes as root, *e.g.*, the ones deriving from $Quality$ will be in *Chunk B*. A similar approach is taken for ontology properties.

Nodes participating in a u-KB will manage a cache of ontology chunks. Every node will have the UO, which is relatively small by design, as well as the chunk(s) required by detained semantic resource annotations. The latter requirement, however, is not strict: if a node has very low storage or bandwidth, it can rely opportunistically on more capable nodes to provide the chunks.

When a node equipped with a reasoner performs service/resource discovery, as a preliminary task it must rebuild a subset of the ontology containing the classes used in the logical expressions of the involved annotations, as well as all their ancestor classes, in order to guarantee correctness of reasoning procedures. To do so, it publishes a message with the *BuildTBox* middleware topic, which all semantic-enabled nodes must be subscribed to. The message

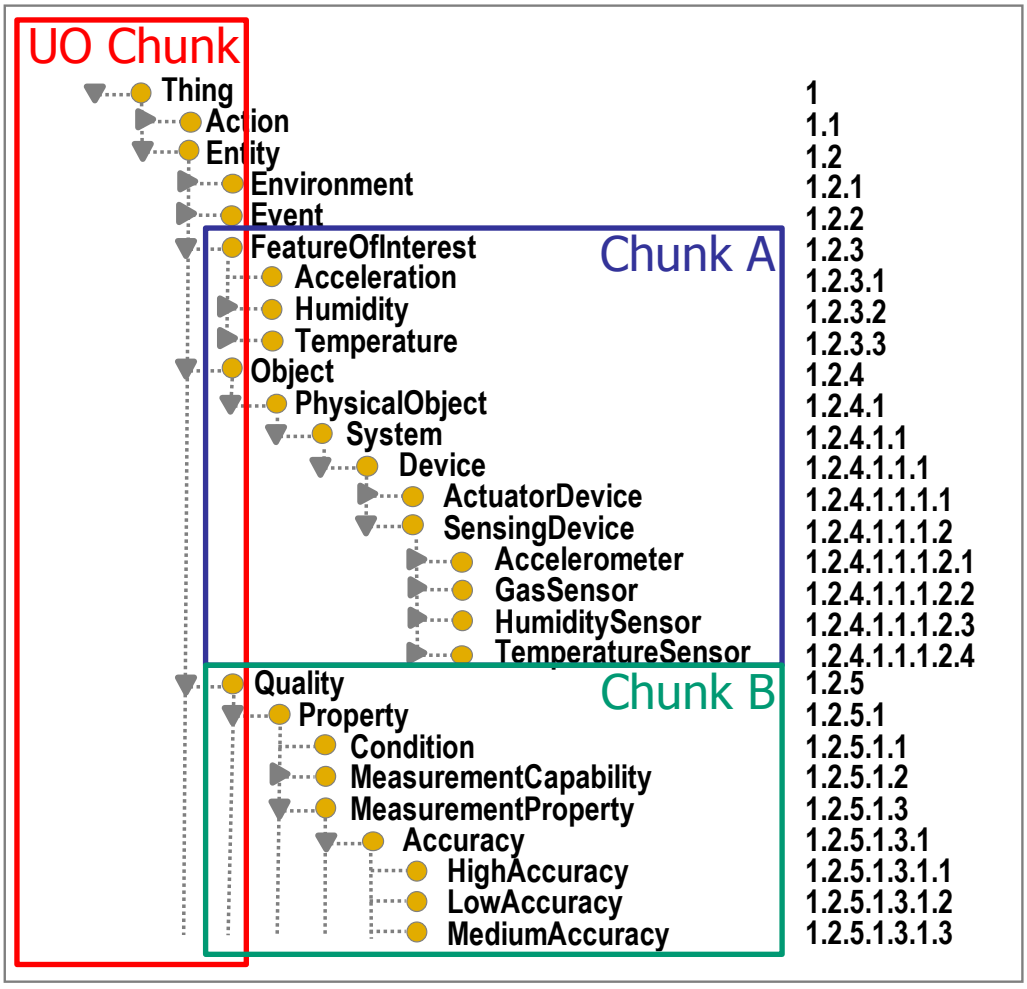


Figure 3.8: Example of ontology excerpt with corresponding class IDs and chunk splitting

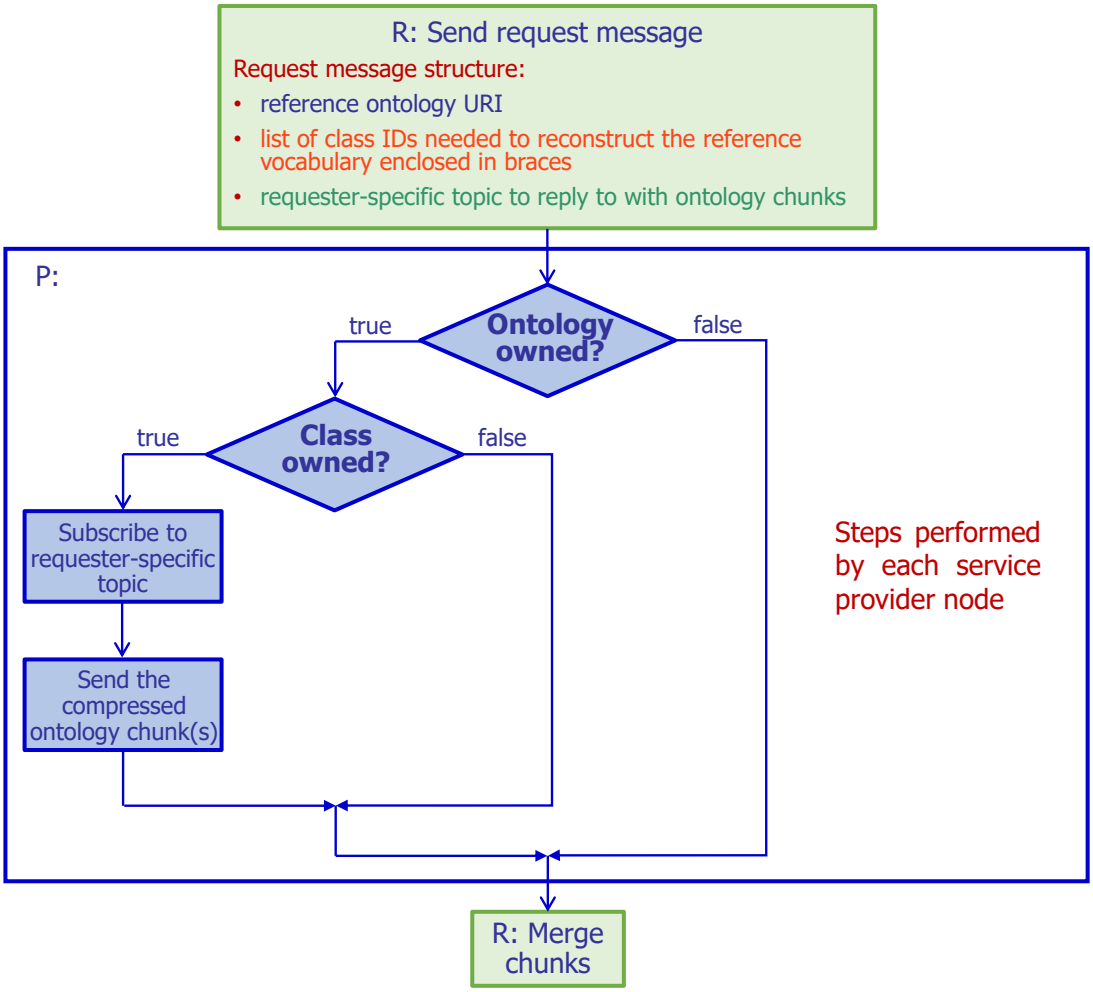


Figure 3.9: Ontology reconstruction process

contains: (i) the unique ontology URI, (ii) the list of requested class IDs, and (iii) the topic name (*e.g.*, *MergeOnto*_NodeID) to be used in reply messages. If a node has one or more requested class IDs in its cache, it will register a publisher on the above topic and send the compressed ontology chunk(s) containing those classes. Merge topics are distinct per requester node to avoid delivering unsolicited chunk copies. After u-KB materialization, the service discovery phase can start. Figure 3.9 summarizes the ontology reconstruction process: the green blocks refer to the requester R, while the blue ones to service providers P.

The proposed approach uses message broadcast in order to send a request to all nodes simultaneously. Messages traveling in broadcast through

the network are very small and contain only meta data, while semantic information, *e.g.*, ontology chunks, are sent in unicast to the requester. This aims to reduce network load, a crucial aspect in pervasive contexts.

3.5.2 Semantic matchmaking for knowledge discovery

The resource/service discovery layer enables the support for dynamic semantic-based discovery of resources scattered in the environment and characterized by semantic annotations. The proposed approach is based on a semantic resource request consisting of an annotation expressed w.r.t. a reference ontology whose URI implicitly defines the application domain of the request. The service discovery phase leverages a general topic named *Discovery*, which all nodes in the network are subscribed to. The requester starts inquiry by sending a *Discovery* message containing: (i) the URI of the ontology the request refers to and (ii) the topic name (by design, *SemAnn*_NodeID) to be used in reply messages; also in this case, the reply topic is node-specific.

Nodes receiving the request check whether they own services/resources related to the same domain. Only in this case, nodes become publishers on the reply topic and send back the related compressed annotations; each annotation is associated with a service-specific topic. The requester collects all descriptions and compares them with its request through semantic matchmaking. The matchmaking process employed here is basically the same described in Figure 3.5 based on non-standard inference services to enable support for approximate matches, resource ranking and formal explanation of outcomes. The outcome of the match determines a ranked list of services/resources which best satisfy the request. In the proposed framework, the overall relevance score of a resource A w.r.t. a request B is computed as:

$$d(A,B) = 100(1 - \frac{penalty^{(c)} + penalty^{(a)}}{penalty^{(a)}_{max}}) \tag{3.6}$$

with terms have the same meaning as in Equation 3.3. The score is converted to an ascending percentage scale for expressing semantic affinity and the requester selects the available service/resource with the highest ranking.

Finally, the requester uses the topic(s) associated to the selected service(s) in order to start fruition. In case of data gathering services, such as from sensors, the requester will act as a subscriber to receive information; on the other hand, controllable resources require the service user to be a publisher on the topic to send commands and data. Figure 3.10 summarizes the Resource allotment process: the green blocks refer to the requester R, while the blue ones to service providers P.

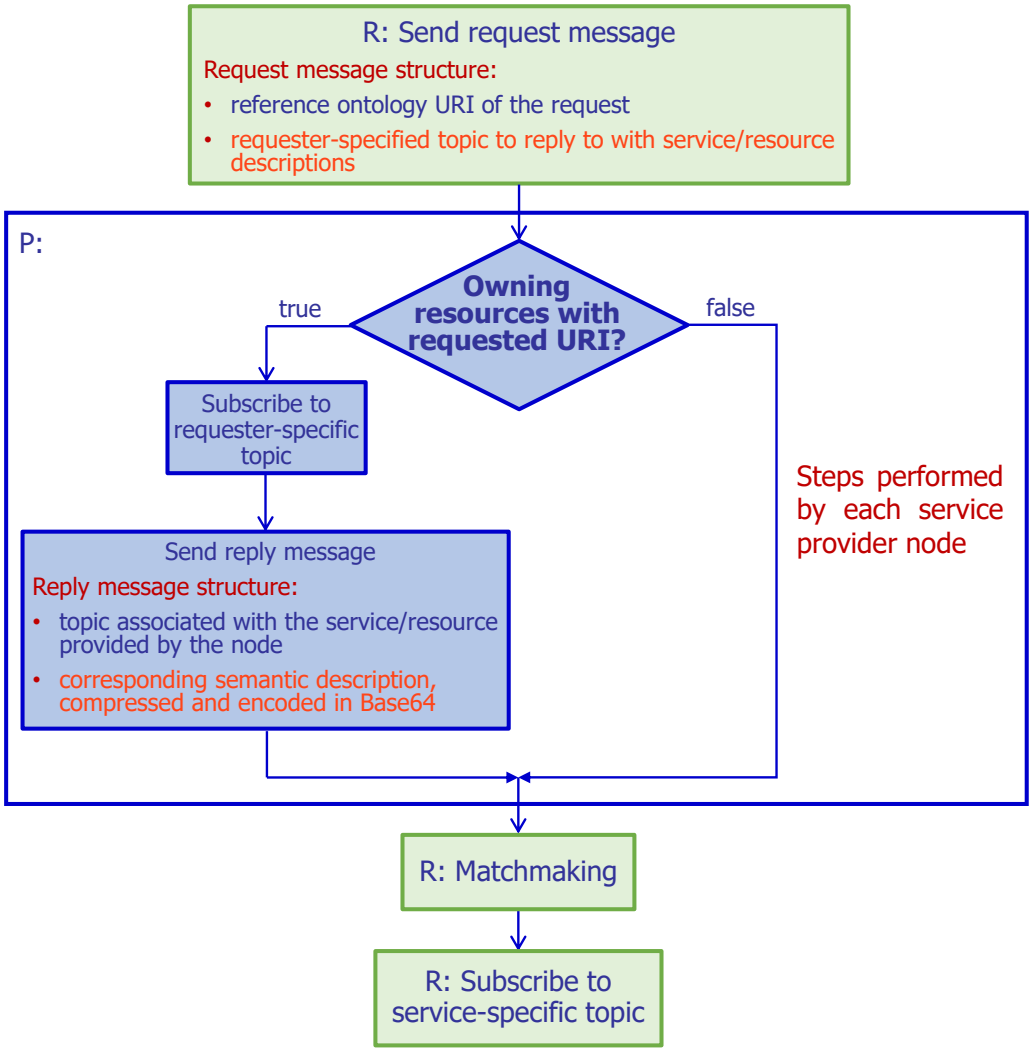


Figure 3.10: Resource allotment process

Also in this case, messages spread in broadcast through the network have reduced size, as they only contain meta data, while semantic annotations which are much more verbose, are sent in unicast to the requester.

In the proposed framework, the above non-standard inferences for semantic matchmaking are implemented in the *Mini-ME* reasoning engine [70], which is suitable for computationally constrained nodes. It works on KBs in the $\mathcal{ALN}$ DL.

Chapter 4

Prototypes and experiments

This chapter provides a detailed description about the prototypical implementation of the proposed framework and an assessment of the dissertation feasibility through experimental results carried out on selected case studies highlighting strengths and limits of the approach. Tests aimed to assess intra- and inter-node performace. Intra-node performance is related to the behavior of the mining and annotation approach adopted by individual smart objects within the network in order to detect and characterize their context. Inter-node performance refers to the evaluation of communication through the devised semantic-enabled middleware infrastructure.

4.1 Context mining

In order to provide an assessment of the feasibility of the context mining proposal, the prototypical implementation and the experimental analysis are descibed in the next sections considering the approach application to a selected case study.

4.1.1 Testbed and implementation

The proposed context mining framework was implemented in a Java-based system prototype to early evaluate its feasibility. The architecture for each smart object consists of three basic modules:

- *Clustering*: input data are clustered invoking the *k-Means* algorithm provided by the *Weka 3.7* library[1] [36] which provides the implementation of machine learning algorithm collection for data mining tasks.

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

- *Advanced k-NN*: this step performs a semantic-enhanced classification of the contextual property observed by the smart object. Inference services are provided by the embedded *Mini-ME 2.0.0*[2] matchmaking and reasoning engine [70].

- *Semantic-based matchmaking*: also this module exploits Mini-ME to infer the environmental state from the semantic-based context description.

In order to evaluate the usefulness of the framework in a real scenario, a prototypical testbed was exploiting an *iRobot Create 2* programmable robot[3] whose characteristics are listed in Table 2.1 of Section 2.2.4. The device was enriched with additional sensors and peripherals. The robot must be able to automatically (b)log a properly annotated description of the context it is located in.

4.1.2 Illustrative example

In *ambient intelligence* [71], indoor spaces –*e.g.*, home or office– become *smart*, *i.e.*, able to learn from their inhabitants, recognize their behavior patterns automatically and provide unobtrusive support for their daily life. The case study proposed here is related to a *home/office smart butler*. It is a robotic agent which exploits embedded as well as external sensors to achieve situation awareness while moving along rooms and corridors. For example, it can interact wirelessly with RFID readers placed at the entryways to detect tagged people who enter/exit. It is then capable of summarizing the gathered information in a semantically annotated description of the environment and its occupants, and publish it to a semantic-enhanced home/building automation infrastructure [64] to schedule other appliances and detect possible issues. Besides, it can autonomously perform several services, such as cleaning, guiding guests toward a particular room or person, or delivering small objects (*e.g.*, paper sheets) across different rooms. In what follows, an illustrative example is presented to better explain the flexibility of the approach.

Mr Brown's house is equipped with a smart refrigerator and a smart pantry, orchestrated by a smart butler which plays the role of coordinator.

The information exchange about the environment state detected by these objects occurs through wireless communication protocols such as IEEE 802.11, Bluetooth or ZigBee. Both the refrigerator and the pantry are able to monitor

[2] `http://sisinflab.poliba.it/swottools/minime/`

[3] `http://www.irobot.com/About-iRobot/STEM/Create-2.aspx/`

Table 4.1: Geometric and Contextual Features for Temperature property

Geometric features	**Contextual features**
– Mean – Variance – Kurtosis – Skewness	– FoodLocation: (*Fridge, KitchenCabinetNearOven, KitchenCabinetUnderSink, etc*) – FoodContainer: (*HermeticContainer, AluminumBox, GlassJar, etc*) – StorageLocationUsage: (*WidelyUsed, RarelyUsed*)

food availability/consumption and control organoleptic qualities of products to detect expired food items. In order to fulfill these tasks estimating food quality degradation level, the dissertation exploits the sensor infrastructure and the mathematical model defined in [4]. The sensor infrastructure consists of temperature sensor, ambient light sensor and relative humidity sensor. Such sensor nodes are situated in kitchen cabinets and in the fridge.

The smart butler exploits the proposed framework to analyze data collected by each sensor and to determine organoleptic property values of food items for the whole observation window. For example, geometric and contextual features that describe the temperature property are reported in Table 4.1. Temperature semantic-based classification value is calculated considering not only quantitative statistical parameters, but also the context components which influence the variation of organoleptic quality. Possible class values are as *VeryLowTemperature*, *LowTemperature*, *MediumTemperature*, *HighTemperature* and *VeryHighTemperature*. For example, the temperature value in a kitchen cabinet is affected by its position: if it is near a switched-on oven, the temperature value within the cabinet may change. Also, a low temperature value inferred for the fridge has not the same meaning of a low temperature value inferred for the freezer. Furthermore, if a food item is stored in a sealed container, the temperature value of the cabinet is less significant. Another important factor that determines the temperature variation in a refrigerator, for example, is related to how many times its door is opened.

Table 4.2 shows the results for temperature property of *rice* stored in the pantry. In detail, advanced k-NN runs with k=7 and assigns the most frequent label among the 7 training samples (TE_i) nearest to the input instance. In particular, with a distance threshold of 50 only the items in red in the table are considered near, therefore the selected class is *LowTemperature* (3/7 *LowTemperature*, 2/7 *VeryLowTemperature*, 1/7 *MediumTemperature* and 1/7 *HighTemperature* nearby points). By replicating this process for

Table 4.2: Temperature property classification with advanced k-NN

Property	**TE**	f_{gs}	f_{cs}	F	**Result**
Temperature	TE_1	0.2500	0.2583	62.82	VeryLowTemperature
	TE_2	0.7084	0.1208	26.58	
	TE_3	0.6667	0.3333	35.33	
	TE_4	0.5417	0.1708	39.80	**LowTemperature**
	TE_5	0.7084	0.0000	17.92	
	TE_6	0.8715	0.0719	13.49	
	TE_7	0.500	0.4833	51.66	MediumTemperature
	TE_8	0.2917	0.3333	63.02	
	TE_9	0.4271	0.1281	44.92	
	TE_{10}	0.3333	0.4332	63.19	HighTemperature
	TE_{11}	0.2917	0.5791	70.04	
	TE_{12}	0.5521	0.5010	47.82	
	TE_{13}	0.2917	0.4208	65.87	VeryHighTemperature
	TE_{14}	0.5000	0.5166	52.33	
	TE_{15}	0.2084	0.3958	71,01	

each sensed parameter, the smart objects (*e.g.*, pantry or refrigerator) create a high-level representation of the considered product conservation status. A food item semantic description detected by the system follows as an example.

$$Rice \sqcap \forall has\_Storage\_Temperature.LowTemperature \sqcap$$
$$\exists\ has\_Storage\_Temperature \sqcap \forall has\_Storage\_Humidity.$$
$$(VeryLowHumidity \sqcap \neg HighHumidity) \sqcap$$
$$\exists\ has\_Storage\_Humidity \sqcap \forall has\_Storage\_Lighting.$$
$$(DimLight \sqcap \neg BrightLight) \sqcap \exists\ has\_Storage\_Lighting$$
$$\sqcap \forall has\_Storage\_Location.Pantry \sqcap \exists has\_Storage\_Location$$

The coordinator butler knows recommended characteristics for the storage of a specific product, hence it performs a second-level matchmaking process to create a list of well-preserved food items. Descriptions of some food items stored in Mr Brown' pantry and descriptions of related recommended characteristics are reported in Table 4.3: semantic matchmaking provides an evaluation of the state of preservation. According to this list as well as to health concerns (*e.g.*, food intolerances, retrieved from semantically annotated user profiles), the butler can suggest the most suitable recipes.

Table 4.3: Semantic-based descriptions of stored food items

Recommended storage description	**Detected storage description**	**Storage status**
$Rice \sqcap$ $\forall has\_Storage\_Temperature.Low\_Temperature \sqcap$ $\forall has\_Storage\_Humidity.Very\_Low\_Humidity \sqcap$ $\forall has\_Storage\_Oxygen.Very\_Low\_Oxygen \sqcap$ $\forall has\_Storage\_Lighting.Dim\_Light \sqcap$ $\forall has\_Storage\_Location.Pantry$	$Rice \sqcap$ $\forall has\_Storage\_Temperature.High\_Temperature \sqcap$ $\forall has\_Storage\_Humidity.Medium\_Humidity \sqcap$ $\forall has\_Storage\_Oxygen.Very\_Low\_Oxygen \sqcap$ $\forall has\_Storage\_Lighting.Bright\_Light \sqcap$ $\forall has\_Storage\_Location.Kitchen\_Cabinet\_01$	Bad
$Potato \sqcap$ $\forall has\_Storage\_Temperature.Low\_Temperature \sqcap$ $\forall has\_Storage\_Humidity.Low\_Humidity \sqcap$ $\forall has\_Storage\_Lighting.Dark\_Light \sqcap$ $\forall has\_Storage\_Location.Closet$	$Potato \sqcap$ $\forall has\_Storage\_Temperature.Low\_Temperature \sqcap$ $\forall has\_Storage\_Humidity.Medium\_Humidity \sqcap$ $\forall has\_Storage\_Lighting.Dark\_Light \sqcap$ $\forall has\_Storage\_Location.Closet$	Good
$Flour \sqcap$ $\forall has\_Storage\_Temperature.Medium\_Temperature \sqcap$ $\forall has\_Storage\_Humidity.Low\_Humidity \sqcap$ $\forall has\_Storage\_Oxygen.Very\_Low\_Oxygen \sqcap$ $\forall has\_Storage\_Lighting.Dark\_Light \sqcap$ $\forall has\_Storage\_Location.Pantry \sqcap$ $\forall has\_Storage\_Container.Hermetic\_Container$	$Flour \sqcap$ $\forall has\_Storage\_Temperature.Low\_Temperature \sqcap$ $\forall has\_Storage\_Humidity.Low\_Humidity \sqcap$ $\forall has\_Storage\_Oxygen.Low\_Oxygen \sqcap$ $\forall has\_Storage\_Lighting.Dark\_Light \sqcap$ $\forall has\_Storage\_Location.Kitchen\_Cabinet\_03 \sqcap$ $\forall has\_Storage\_Container.Hermetic\_Container$	Good

4.1.3 Tests and results

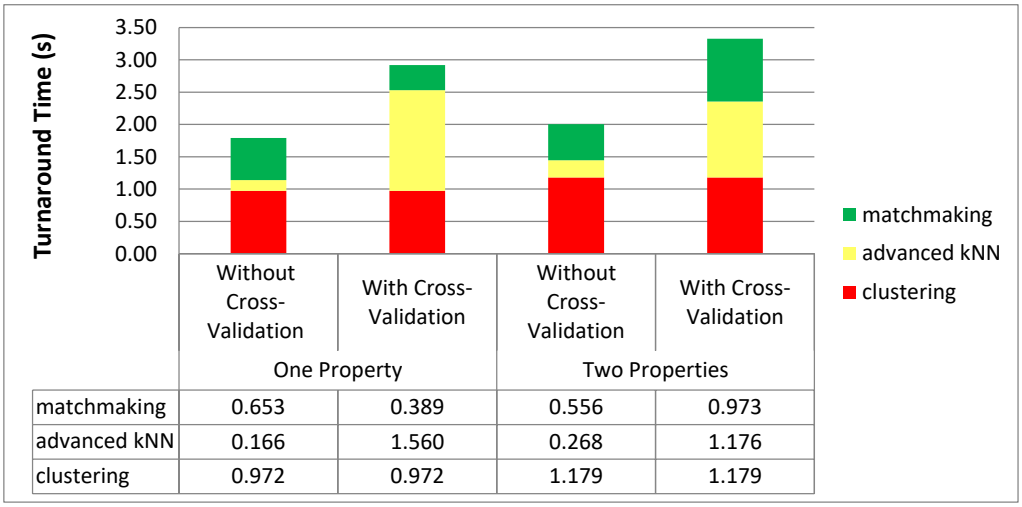
Experimental analysis of the proposed semantic-based framework was conducted on a dataset of 400 real instances of weather sensor data (temperature and humidity, collected from Weather Underground[4] Web service) to simulate sensor data gathering by a smart object. The training set exploited for testing the proposal is populated by 15 temperature samples and 9 humidity ones. The tests were performed in two different conditions: with static value of k for the *advanced k-NN* phase ($k = 5$ for temperature detection and $k = 3$ for humidity one) and with cross validation (useful to set k dynamically). The matchmaking task works with an ontology whose size is 221 kB. As a result of a series of system executions to tune configuration variables modeling the *advanced k-NN* approach, values for the parameters in the score combination function (explained in Section 3.3.1) were set as follows: $\alpha = 0.8$, $\gamma = 0.02$, $\epsilon = 0.02$ and $\delta = 0.9$. As performance metrics, turnaround time of data point processing and RAM usage were considered for each module of the proposed mining framework.

Preliminary performance evaluation was carried out on a PC testbed equipped with an Intel Core i7 Q720 CPU, 1.60 GHz clock, 4 GB RAM, 64-bit Windows 7 Professional operating system and 64-bit Java 7 SE Runtime Environment (build 1.7.0_67-b01). Time was measured through timestamping instructions embedded in the source code. Each test was repeated 10 times and the average values were taken. Figure 4.1 reports turnaround time results for the analysis of only one and both properties, with and without cross validation. As expected, turnaround time increases significantly when the system performs cross validation to set the best k value for k-NN. The most significant differences between results for one and two properties are in the clustering and matchmaking phases, but the time increase is less than linear.

For memory usage analysis, the Java Memory Monitor profiler in the Eclipse platform was used. In this case, reported results are the peaks of 10 runs. They are shown in Figure 4.2. The memory usage for a single observed sensor property is almost equal to the one for two properties or more. RAM occupancy is always below 14 MB. Overall memory peaks correspond to the most data intensive tasks, *i.e.*, cross validation and matchmaking.

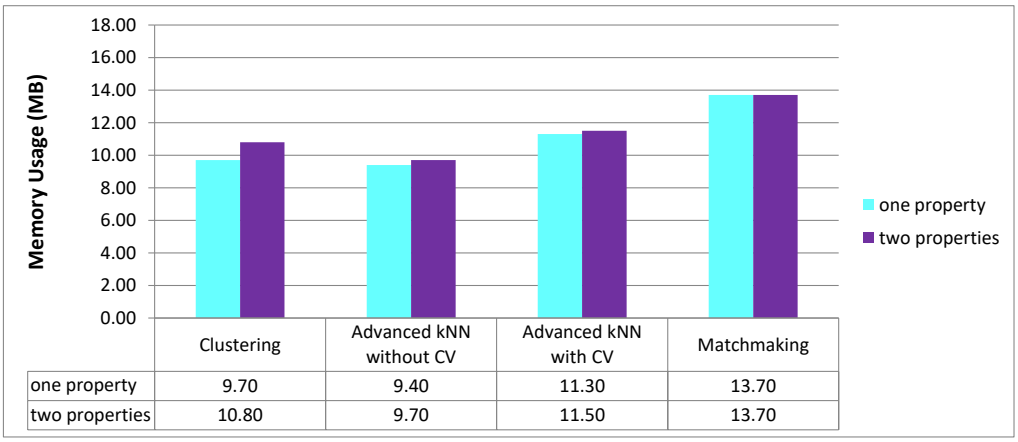
The proposed semantic-based framework was designed for advanced event identification in mobile and pervasive computing. Those contexts are characterized by severe resource limitations affecting processing, memory, storage and energy consumption. Therefore, hardware and software limitations should be taken into account during system evaluation. In order to evaluate

[4] `http://www.wunderground.com/`



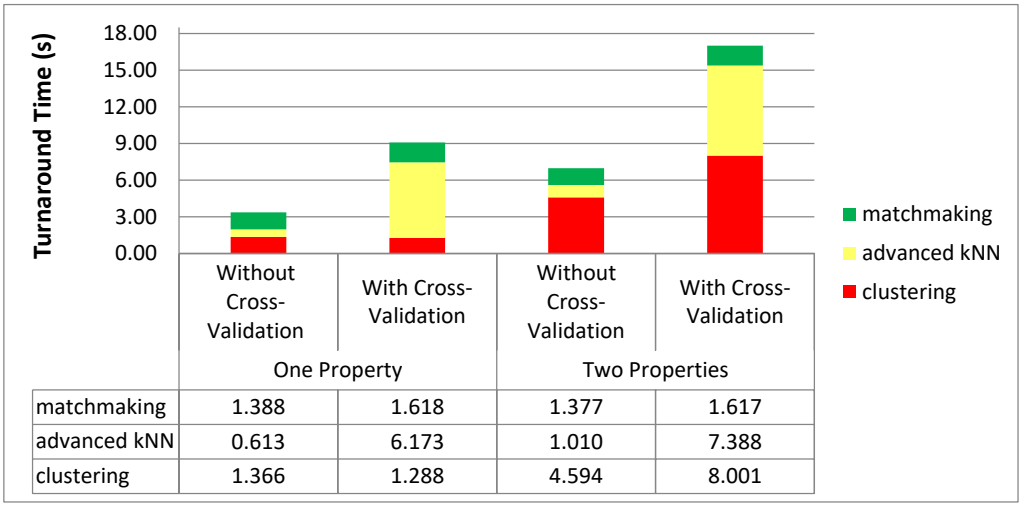
	One Property		Two Properties	
	Without Cross-Validation	With Cross-Validation	Without Cross-Validation	With Cross-Validation
matchmaking	0.653	0.389	0.556	0.973
advanced kNN	0.166	1.560	0.268	1.176
clustering	0.972	0.972	1.179	1.179

Figure 4.1: Turnaround time results on PC testbed



	Clustering	Advanced kNN without CV	Advanced kNN with CV	Matchmaking
one property	9.70	9.40	11.30	13.70
two properties	10.80	9.70	11.50	13.70

Figure 4.2: Memory usage on PC testbed



	One Property		Two Properties	
	Without Cross-Validation	With Cross-Validation	Without Cross-Validation	With Cross-Validation
matchmaking	1.388	1.618	1.377	1.617
advanced kNN	0.613	6.173	1.010	7.388
clustering	1.366	1.288	4.594	8.001

Figure 4.3: Turnaround time results on Raspberry Pi

a realistic smart object platform with limited resources, further performance evaluation was carried out on a Raspberry Pi single-board computer equipped with Broadcom BCM2835 system on a chip, with an ARM1176JZF-S CPU at 700 MHz, 512 MB RAM and an SD card for booting and long-term storage. It was configured with Wheezy Debian GNU/Linux 7.8 operating system, Wi-Pi WLAN USB Module[5] and 32-bit Java 7 SE Runtime Environment (build 1.8.0-b132).

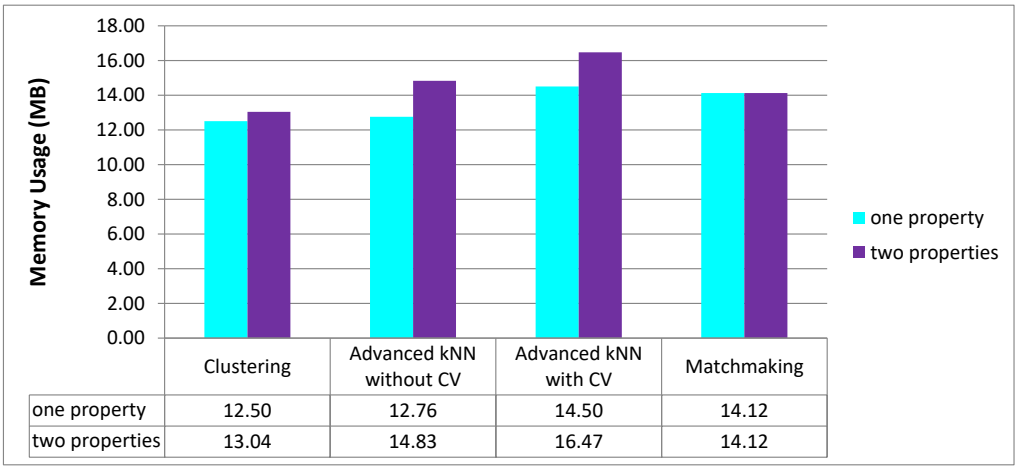
Figure 4.3 reports turnaround time results for the analysis of only one and both properties, with and without cross validation. Turnaround time is about three times higher compared to the PC testbed, but the relative duration of each processing step is quite similar.

For memory usage analysis, *jvmtop* tool[6] was used to profile memory usage at runtime. Test results are shown in Figure 4.4. RAM occupancy is always below 17 MB and also in this case, memory peaks correspond to the most data intensive tasks, *i.e.*, cross validation and matchmaking. Memory usage is higher with respect to that obtained from the preliminary tests carried out on the PC testbed. This is likely due to the smaller storage capabilities of Raspberry Pi.

These preliminary tests evidence that the proposed framework provides acceptable results, even though optimizations are needed to run in realistic scenarios on resource-constrained computing platforms.

[5] `http://it.farnell.com/element14/wipi/dongle-wifi-usb-for-raspberry-pi/dp/2133900`

[6] `https://code.google.com/p/jvmtop/`



	Clustering	Advanced kNN without CV	Advanced kNN with CV	Matchmaking
one property	12.50	12.76	14.50	14.12
two properties	13.04	14.83	16.47	14.12

Figure 4.4: Memory usage on Raspberry Pi

4.2 Semantic-enhanced middleware for knowledge discovery and allotment

In order to asssess the benefits of the proposed knowledge discovery and sharing approach, a software prototype has been implemented and performance evaluations were carried out with reference to a case study in the smart agriculture field.

4.2.1 Testbed and implementation

The semantic layers with related functionality described in Section 3.5 were implemented in *Bee-DDS*[7], a middleware infrastructure providing Data Distribution Services in real-time through a publish-subscribe topic-based protocol to regulate data exchange among system nodes by implementing the Object Management Group (OMG) specifications[8]. Each topic is defined by two components: (i) a *data type*, which describes the data organization of the topic, and (ii) a *name* which unambiguously identifies the topic within a Bee Domain. In the Java-based off-the-shelf Bee-DDS platform, service discovery is carried out via basic string-matching of topics. The implementation of the semantic layers proposed here aims at making it more flexible and meaningful. Matchmaking in supported by the embedded Mini-ME 2.0[9]

[7]Bee Data Distribution System, `http://sine.ni.com/nips/cds/view/p/lang/it/nid/211025/`

[8]DDS specifications, `http://www.omg.org/spec/DDS/Current`

[9]`http://sisinflab.poliba.it/swottools/minime/`

reasoning engine [70], providing non-standard inference services described in Section 2.3.3. The novel semantic layers were implemented in Java language and integrated in the preexisting software infrastructure.

In order to evaluate the usefulness of the framework, a prototypical testbed was developed exploiting a semantic sensor network and a 3D Robotics Iris$^+$ drone (whose features are shown in Table 2.1 of Section 2.2.4) equipped with additional sensors and peripherals. The cooperation of these entities is achieved by means of the support provided by the semantic-enhanced middleware infrastructure proposed in this thesis.

Automatic reasoning system on Pixhawk

3D Robotics Iris$^+$ quadcopter includes the advanced Pixhawk autopilot system described in Table 2.1 of Section 2.2.4.

The aim of this project is to confer to the drone the ability to perform knowledge discovery via semantic matchmaking through a dedicated software module to be integrated within the Pixhawk autopilot. As said above, this task is achieved by leveraging the embedded Mini-ME 2.0 reasoning engine for mobile and embedded devices, written in Java language.

Since no Java Virtual Machine implementation is available for the NuttX RTOS running on Pixhawk, a porting effort was carried out.

The starting point was the *Kilobyte Virtual Machine* (KVM)[10]. The main KVM features are: (i) compact size of the VM (50-80 kilobytes); (ii) reduced memory usage; (iii) satisfactory performance and (iv) portability.

Figure 4.5 and Figure 4.6 show the pictures of the experimental testbed. The laptop - Pixhawk device communication took place via a serial connection using a USB-TTL interface cable.

Starting from the existing implementation of KVM for Unix and the build configuration for the Linux environment, the corresponding KVM for NuttX was built by performing a cross-compilation of the application through the use of NuttX firmware in Unix environment. The croos-compilation produced a binary file which could be uploaded directly on the Pixhawk machine. The NuttX firmware supported by Pixhawk is *APM:Copter*[11] (formerly called ArduCopter).

Cross-compilation required the replacement of required standard Linux libraries with the corresponding libraries that implement the same functionality in the NuttX environment.

[10]KVM is open source, so the source code is available on `http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javame-419430.html#J2MECLDC-1.1-WINUNIX-G-F`

[11]Firmware installation guide, `http://dev.px4.io/index.html`

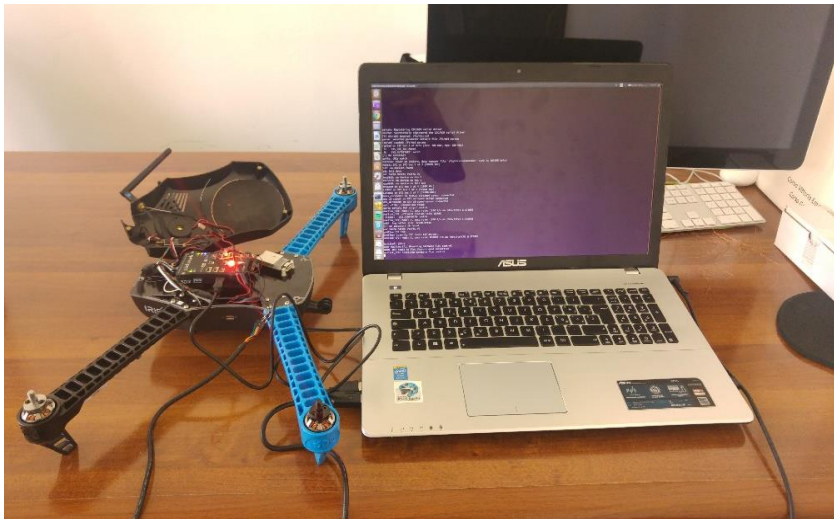


Figure 4.5: Experimental testbed

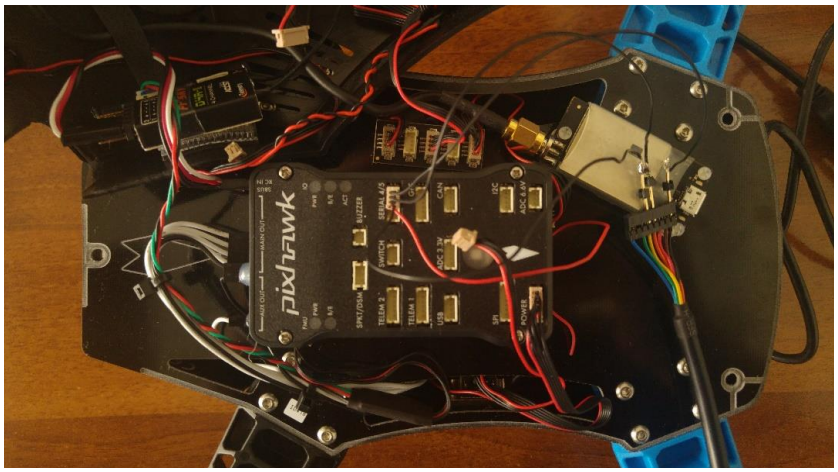


Figure 4.6: Detail of experimental testbed connections

4.2.2 Illustrative example

The clarifying example proposed here is related to the smart agriculture field, where objects can share information in order to monitor crops by means of appropriate sensors for finding out whether it is necessary to irrigate or spray with pesticides the field. In this context, the knowledge sharing system plays a fundamental and indispensable role in order to enable cooperation of the involved entities allowing to detect the specific field context state, formulate plans to reach the mission goals and act accordingly. In what follows, an illustrative example is presented to clarify functional and non-functional aspects of the thesis.

Downy mildew is a serious fungal disease of grapevine which can result in severe crop loss. It is caused by the fungus Plasmopara viticola. *The pathogen attacks all green parts of the vine, especially the leaves. In order to eliminate the fungus, a smart vine monitoring is realized by analyzing environmental parameters collected by a sensor network. According with this monitoring, a smart farming drone is able to automatically infer when, where and how spraying fungicides on susceptible cultivars.*

Environmental factors influencing the development of *Plasmopara viticola* include relative humidity, atmospheric pressure, soil moisture, leaf wetness, rugged soil temperature, sun calibration quantum, meteorological data. Raw data are collected through the developed semantic-enhanced middleware and processed on the fly by the smart farmer drone. In detail, five nodes are connected to the plant control agriculture software platform:

- **N1**: soil temperature sensor;
- **N2**: relative humidity sensor;
- **N3**: tractor;
- **N4**: soil temperature sensor;
- **N5**: farmer drone.

As shown in Figure 4.7, each node includes a Publisher for data dissemination, through one or more Data Writer (DW) objects and a Subscriber for data gathering through one or more Data Reader (DR) objects, each associated to one Topic subscription.

N5 acts as a requester of knowledge about the environment in order to act on it. When the systems starts, N5 acts a publisher on general topics *BuildT-Box* and *Discovery* and as subscriber on node-specific topics *MergeOnto_N5* and *SemAnn_N5*. It also embeds the Mini-ME logic-based matchmaker [70]

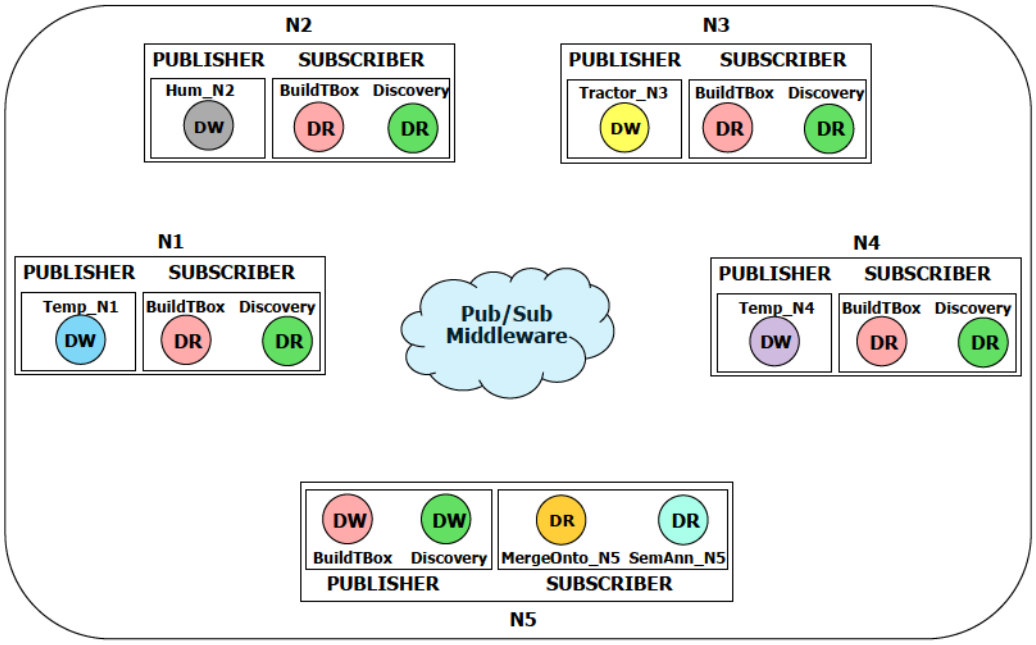


Figure 4.7: Case study: initial system state

for ranking discovered services/resources. On the other hand, N1, N2, N3, N4 are distributed in the monitored area and play the role of service/resource providers. They all subscribe to general topics *BuildTBox* and *Discovery*; furthermore, each provided service has a specific topic associated via the respective node's Publisher (*Temp_N1*, *Hum_N2*, *Tractor_N3* and *Temp_N4*). The knowledge discovery process is composed by the following interaction steps:

Ontology rebuilding

1. As reported in the first line of Table 4.4, N5 requires a soil temperature service, with high accuracy and precision, low measurement range and frequency, and high response time. It composes a semantic-based request according to the ontology reported in Section 3.5.1 (Figure 3.8). Before starting service discovery, it needs to rebuild a minimal self-contained subset of the reference domain terminology, so it posts the following request on the *BuildTBox* topic:

   ```
   http://www.example.com/ontology.owl
   {1.2.5.1.3.7.1;1.2.4.1.1.1.2.4;1.2.3.3;
   1.2.5.1.3.6.2;1.2.5.1.3.9.1;1.2.5.1.3.4.2;
   1.2.5.1.3.1.1}MergeOnto_N5
   ```

   In the request there are: (i) the URI of the OWL2 reference ontology

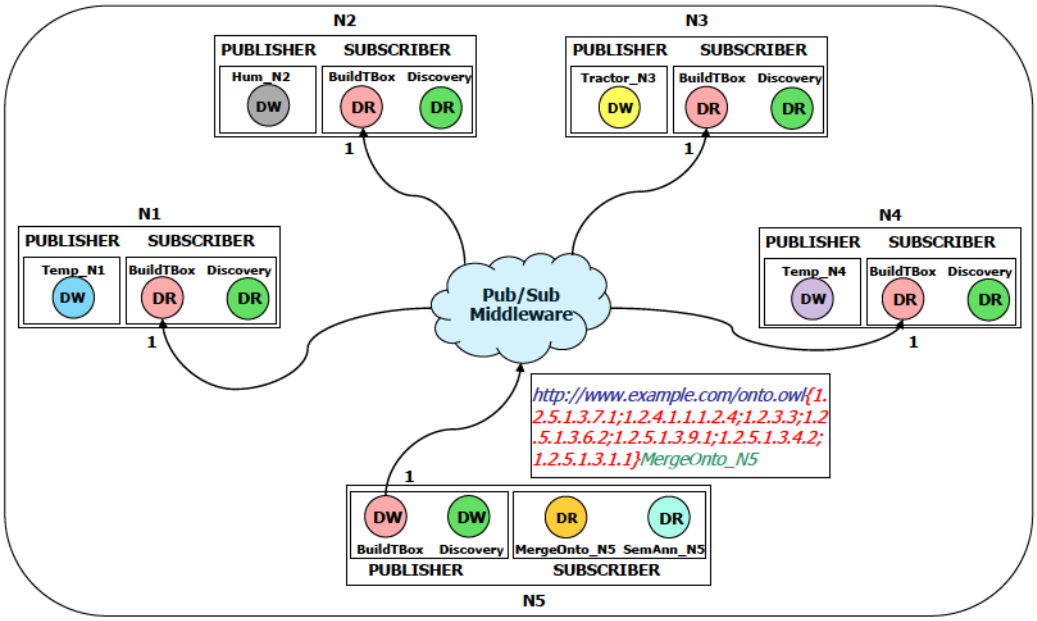


Figure 4.8: Case study: ontology reconstruction - sending request

the request refers to, (ii) the list of class IDs needed to rebuild the needed vocabulary subset, and (iii) the requester-specific topic where nodes will create a DW to reply with ontology chunks (See Figure 4.8).

2. Through the DR on the *BuildTBox* topic, N1, N2, N3 and N4 receive the metadata and check whether the URI in the request refers to (chunks of) an ontology they own. If it does, they verify the correspondence of at least one item in the list of their ontology chunk(s). In that case, a DW on MergeOnto_N5 is created on-the-fly (dynamically created DWs and DRs are shown with a dashed outline in Figure 4.9) for sending selected chunk(s). In the case study example, N1, N2 and N4 reply, whereas N3 does not manage the requested ontology.

3. Through the DR on *MergeOnto_N5*, N5 receives the needed ontology chunks and merges them. The result is a semantically consistent subset of the full ontology, which is enough to infer upon the subsequent request (See Figure 4.9).

Semantic-based service/resource discovery

4. Now N5 can publish the annotation of the requested soil temperature detection service on the *Discovery* topic:

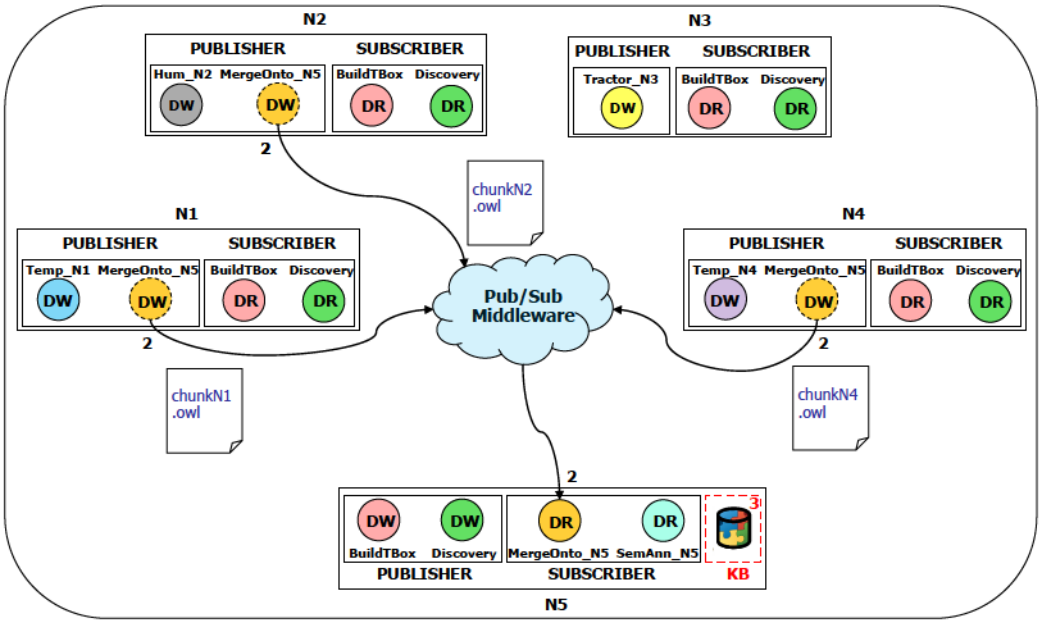


Figure 4.9: Case study: ontology reconstruction - sending and merging chunks

`uri=http://www.example.com/ontology.owl;`
`semanticTopic=SemAnn_N5`

The string is the concatenation of: (i) the URI of the ontology the request refers to and (ii) the requester-specified topic where an interested node has to send back service/resource descriptions (See Figure 4.10).

5. Thanks to *Discovery* topic subscription, N1, N2, N3 and N4 receive the metadata in the request and carry out a (syntactic) match of ontology URIs to ascertain whether they own related services or resources. In the example, N3 has no services described by the specified vocabulary, while the check succeeds for N1, N2 and N4, which become publishers on *SemAnn_N5*. As an example, N1 replies as in what follows:

   `topic=Temp_N1;semanticAnnotation=[...]`

   Each reply string is the concatenation of: (i) the topic associated with the service/resource provided by the node and (ii) the corresponding semantic description, compressed and encoded in Base64 for compatibility purposes (See Figure 4.11).

6. N5 receives the messages of N1, N2 and N4 and executes the matchmaking process between the annotated request and the semantic de-

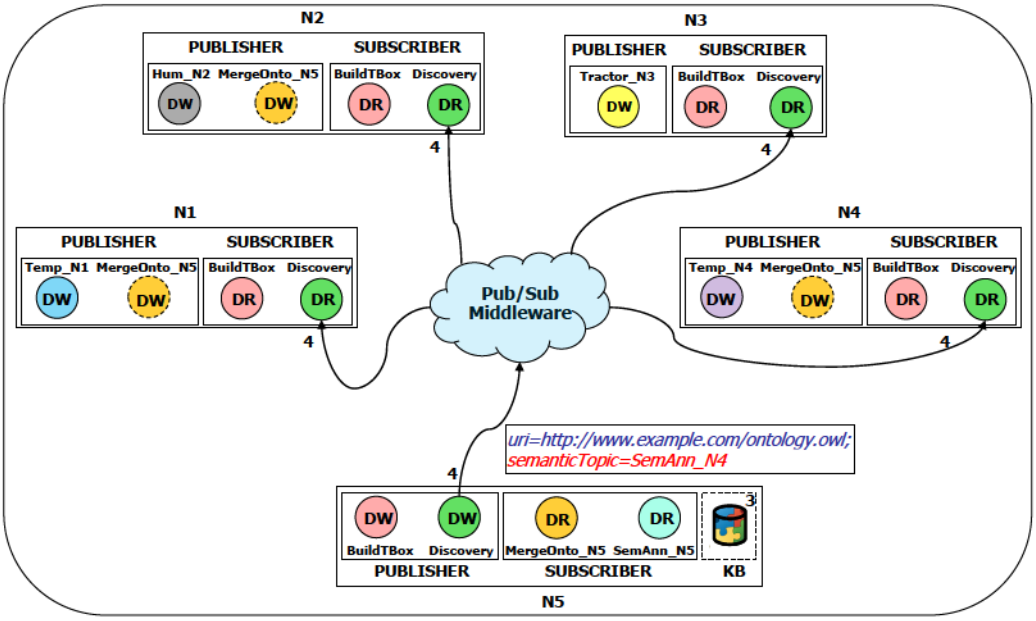


Figure 4.10: Case study: resource allotment - sending request

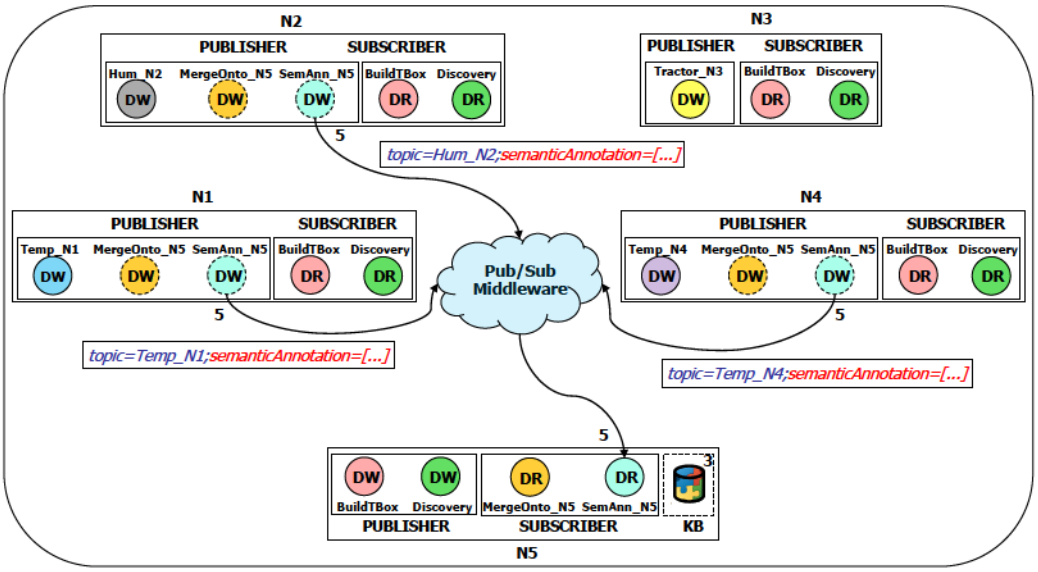


Figure 4.11: Case study: resource allotment - sending service descriptions

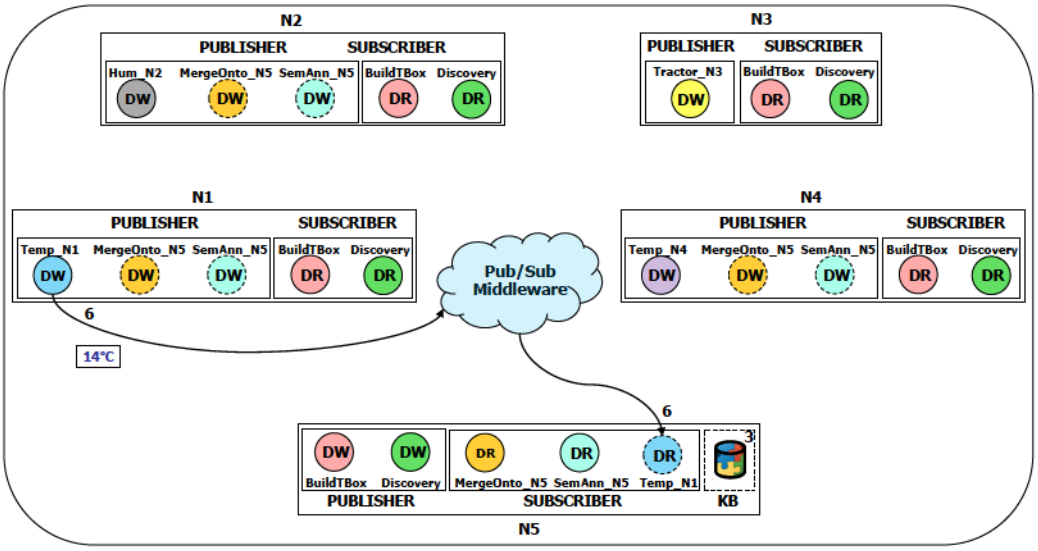


Figure 4.12: Case study: resource allotment - temperature service fruition

scriptions of discovered services. All logic-based descriptions and the matchmaking outcomes are presented in Table 4.4. The best match is achieved by N1 (lowest semantic distance w.r.t. the request); N3 is also close, but more constraints are not matched exactly, while N2 is less relevant as a sensor, because its observed quantity is incompatible. Finally N5 becomes subscriber on *Temp_N1* topic in order to start receiving soil temperature data from the sensor exposed by N1 (See Figure 4.12).

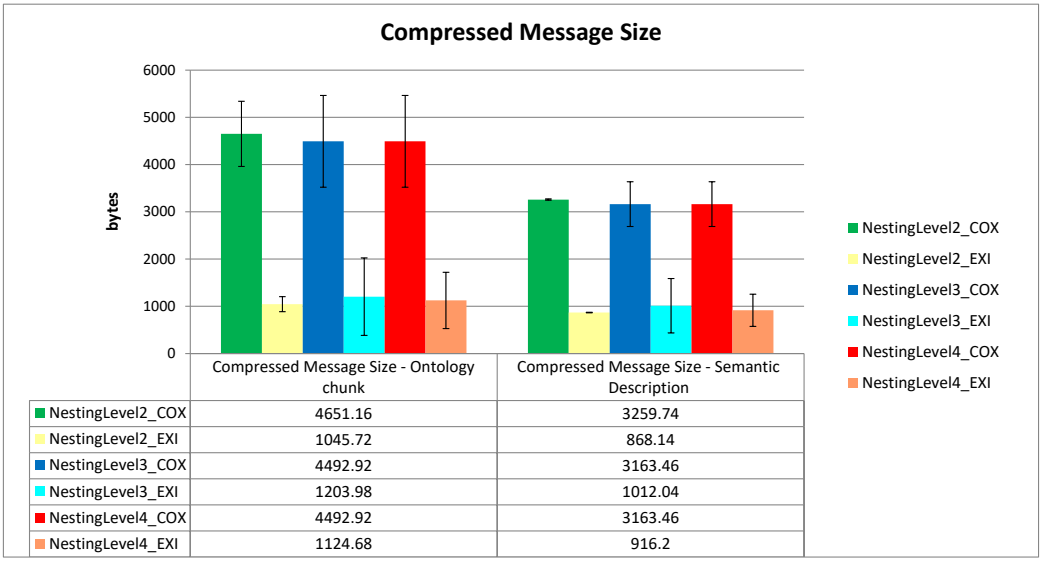
The example was intentionally kept simple for explanatory purposes. In real scenarios, a device could expose more services, possibly belonging to different domains and described through different ontologies. Nevertheless, the basic interaction sequence and mechanisms is the same.

4.2.3 Tests and results

Inter-object network performance was analyzed to evaluate the efficiency and feasibility of the proposed semantic-enhanced middleware platform for knowledge discovery and allotment. Preliminary tests were performed in a small scenario with 50 provider nodes and a single requester node interconnected through the Bee-DDS middleware extended with the proposed semantic layers. Such early evaluation was basically carried out in order to select two system configuration variables: (i) the compression algorithm for ontology chunks and service/resource annotations; (ii) the nesting level of the up-

Table 4.4: Matchmaking outcome

	Node	Semantic Description	Score
Requester	**N5**	$SoilTemperatureSensor \sqcap \forall observes.SoilTemperature \sqcap \forall hasMeasurementProperty.(HighAccuracy \sqcap LowMeasurementRange \sqcap LowFrequency \sqcap HighPrecision \sqcap HighResponseTime)$	n.a.
Provider	**N1**	$SoilTemperatureSensor \sqcap \forall observes.SoilTemperature \sqcap \forall hasMeasurementProperty.(HighAccuracy \sqcap LowFrequency \sqcap MediumMeasurementRange \sqcap HighPrecision \sqcap MediumResponseTime \sqcap MediumResolution \sqcap LowLatency)$	92.6
	N4	$SoilTemperatureSensor \sqcap \forall observes.SoilTemperature \sqcap \forall hasMeasurementProperty.(LowAccuracy \sqcap LowFrequency \sqcap LowMeasurementRange \sqcap LowPrecision \sqcap MediumResponseTime \sqcap LowResolution \sqcap LowLatency)$	85.2
	N2	$RelativeHumiditySensor \sqcap \forall observes.RelativeHumidity \sqcap \forall hasMeasurementProperty.(LowAccuracy \sqcap LowFrequency \sqcap LowMeasurementRange \sqcap MediumPrecision \sqcap MediumResponseTime \sqcap LowResolution \sqcap LowLatency)$	71.4



	Compressed Message Size - Ontology chunk	Compressed Message Size - Semantic Description
NestingLevel2_COX	4651.16	3259.74
NestingLevel2_EXI	1045.72	868.14
NestingLevel3_COX	4492.92	3163.46
NestingLevel3_EXI	1203.98	1012.04
NestingLevel4_COX	4492.92	3163.46
NestingLevel4_EXI	1124.68	916.2

Figure 4.13: Compressed message size (50:1 test)

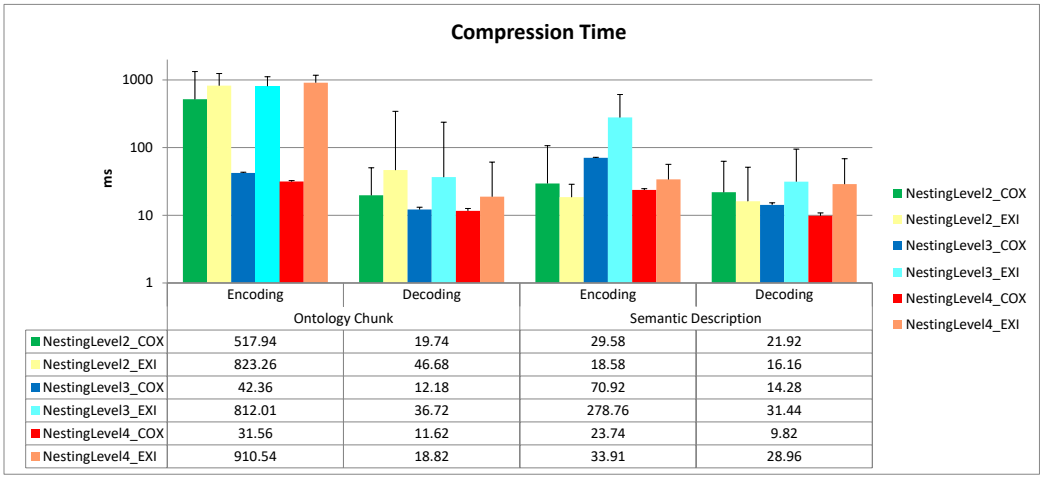
per ontology chunk (UO), shared by all nodes. Compression techniques are fundamental in pervasive and advanced scenarios involving mobile devices featured by limited computational and storage resources, so COX [69] and EXI[12] were considered as encoding algorithms in this experimental software prototype. Furthermore 2, 3 and 4 were taken as possible values of the top-most level in the class hierarchy (see Section 3.5.1).

Performance evaluation was carried out on a PC equipped with Intel Core i7 Q720 CPU at 2.80 GHz, 8 GB RAM, 64-bit Windows 7 Professional operating system and 32-bit Java 8 SE Runtime Environment (build 1.8.0 72-b15). Compressed size of messages and turnaround time for encoding and decoding were considered as performance metrics, both for ontology rebuilding and resource allotment.

Figure 4.13 reports on compressed message size results: EXI performs better than COX in terms of compressed message size. Compression time was measured through timestamping instructions within the source code: results reported in Figure 4.14 show that COX is faster than EXI for both the processing phases.

For subsequent tests, EXI was selected as data compression algorithm. Furthermore, the upper ontology nesting level was set to 4, in accordance with the best trade-off between the desired UO detail level and the time required for message compression and decompression.

[12]Efficient XML Interchange-EXI, `https://www.w3.org/TR/exi/`



	Ontology Chunk		Semantic Description	
	Encoding	Decoding	Encoding	Decoding
NestingLevel2_COX	517.94	19.74	29.58	21.92
NestingLevel2_EXI	823.26	46.68	18.58	16.16
NestingLevel3_COX	42.36	12.18	70.92	14.28
NestingLevel3_EXI	812.01	36.72	278.76	31.44
NestingLevel4_COX	31.56	11.62	23.74	9.82
NestingLevel4_EXI	910.54	18.82	33.91	28.96

Figure 4.14: Time for message encoding and decoding (50:1 test)

After the above preliminary tests, performance was evaluated considering 500 resource providers and 10 requesters. Provider nodes were equally distributed among 50 virtual machines, which were run on different host PCs. Each virtual machine was equipped with dual-core CPU, 800 MB RAM, 32-bit Ubuntu 14.04 LTS operating system and 32-bit Java 8 SE Runtime Environment (build 1.8.0 72-b15). On the other hand, requester nodes were deployed on 3 virtual machines equipped with dual-core CPU, 2 GB RAM, the same operating system and Java runtime environment. The aim was to simulate a more realistic workload, where several nodes located on different computers are connected through the middleware. In the 500:10 scenario tests, host machines were connected to a 100 Mb/s IEEE 802.3 network. Turnaround time and RAM usage were analyzed as performance metrics for the whole discovery process.

Figure 4.15 reports on turnaround time results for the ontology rebuilding phase. The total execution time taken for reassembly a subset of the ontology containing the classes used for reasoning was about 22.67 s on average.

Figure 4.16 shows turnaround time results for resource allotment. This stage took an average total time of 20.2 seconds. For both phases, message transmission was the longest sub-task. This may have been influenced by the middleware configuration and the communication network physical properties. Overall, the service discovery process required less than 41 s in a moderately complex realistic scenario, where 10 applicants sent requests to 500 providers in order to find the best available services/resources. Turnaround time for the the entire system execution was significantly higher than basic MOM middleware, but the proposed approach enables advanced scenarios

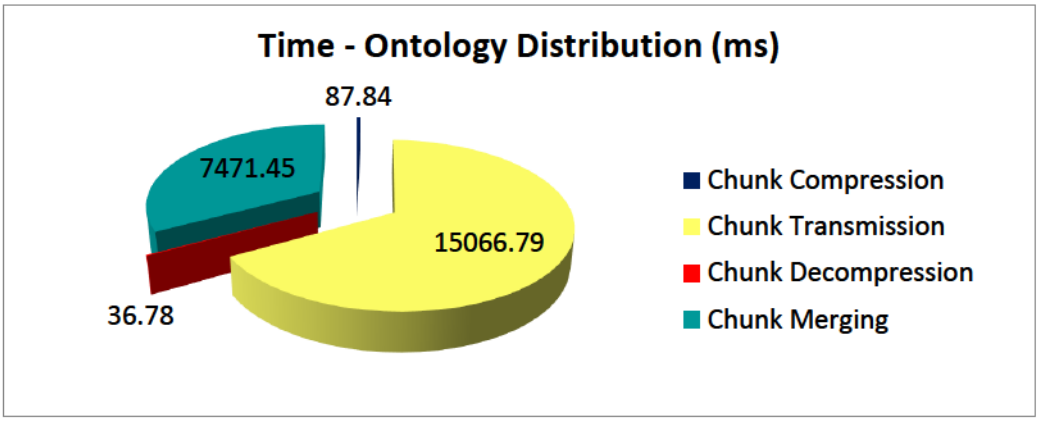


Figure 4.15: Time for ontology rebuilding (500:10 test)

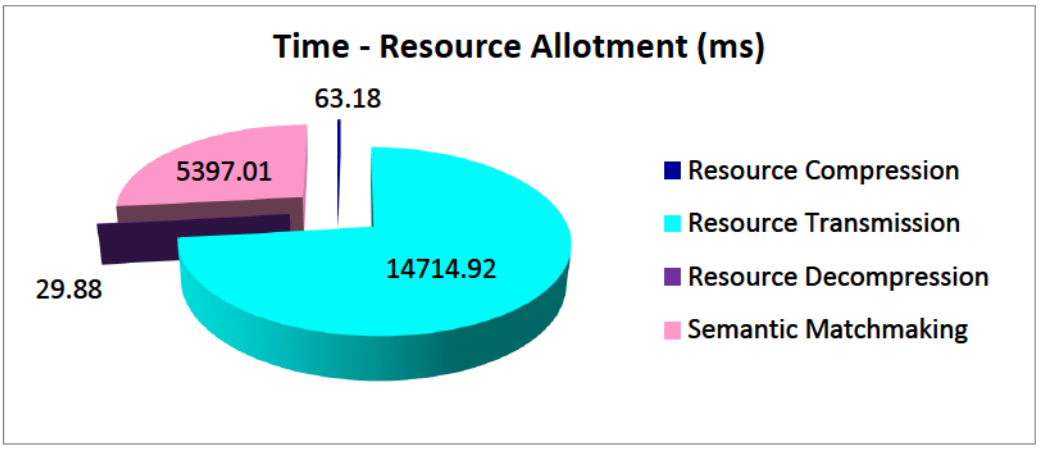


Figure 4.16: Time for resource allotment (500:10 test)

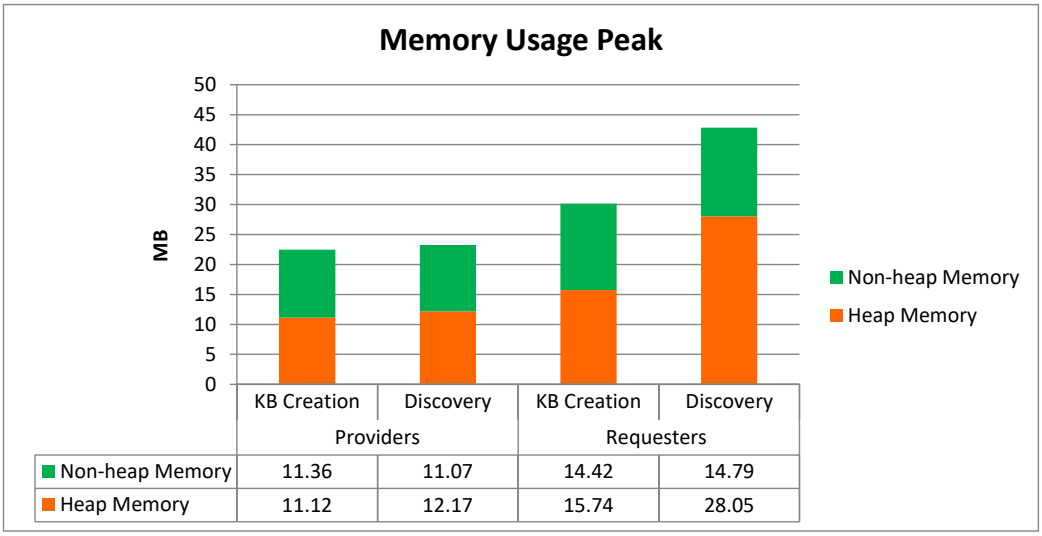


Figure 4.17: Memory usage peak (500:10 test)

where managing complex information is required.

For memory usage analysis, an embedded thread was used to profile RAM at run-time for KB creation as well as for discovery. Results are shown in Figure 4.17: RAM occupancy for provider nodes is lower than for requesters, since the latter are equipped with the reasoner to execute service matchmaking. Memory usage peak for requesters was always below 30 MB for KB creation and approximately 43 MB for the discovery proceess.

Overall, the above tests evidence the feasibility of the proposed middleware enrichment with the support for dynamic semantic-based service/resource discovery. It is also evident some performance optimizations and adjustments are needed.

Chapter 5

Conclusions and perspectives

This thesis introduced the *Object (b)logging* paradigm, defining a novel knowledge based framework for pervasive contexts. The main goal is to enable a smart object to process retrieved data streams in order to describe itself and the environment where it is located. k-NN supervised machine learning algorithm was combined with including non-standard semantic-based reasoning services in order to produce a rich and meaningful semantic representation of events starting from a low-level statistical analysis of data. Most interestingly, dissertation allows knowledge sharing in distributed systems, particularly targeted toward scenarios including large numbers of resource-constrained nodes, by means of a semantic-enhancement layer added on top of an off-the-shelf publish/subscribe middleware. It enables logic-based annotation of both available resource descriptions and requests, as well as dynamic discovery via deductive matchmaking, supporting approximate matches and service ranking by affinity with requests. The thesis includes ubiquitous KB management, with ontology partitioning across multiple nodes and dynamic on-demand rebuilding of the subset just needed for reasoning on a given set of annotations. The proposed solution results as a general-purpose, cross-domain semantic-based context mining, knowledge discovery and sharing facilitator among pervasive smart devices, providing the means to harness the flow of semantically annotated updates inferred from low-level data, enabling context-aware adaptive behaviors in several application areas, including urban search and rescue, personal assistance, home automation, smart agriculture and many more.

The approach was implemented in a working prototype, embedding a semantic matchmaker suitable for mobile settings. Experimental evaluation was carried out in order to assess effectiveness, correctness and feasibility of the proposal with reference to a possible exploitation on resource-constrained platforms. It was applied in challenging pervasive computing case studies

where multi-object teams cooperate by self-coordination for executing tasks or making interventions on the surrounding environment according to context state detection. In order to evaluate the usefulness of the framework, a prototypical testbed was developed exploiting a 3D Robotics Iris$^+$ drone and an iRobot Create 2 programmable robot enriched with additional sensors and devices.

Future work directions concern further performance optimization and comparison with state-of-the-art approaches. Several future perspectives are open for semantic-enhanced machine learning. A proper extension of the baseline training algorithm can enable a continuously evolving model through a fading mechanism to allow the system to "forget" the oldest training samples. Furthermore, adopting a more expressive logic language such as $\mathcal{ALN}(\mathrm{D})$ to model the domain ontologies could allow to introduce data-type properties to better characterize data features. Finally, additional developments related to the knowledge sharing infrastructure include the definition of novel dynamic approaches for service/resource clustering, composition, substitution and requester-provider negotiation.

Bibliography

[1] OWL 2 Web Ontology Language, https://www.w3.org/TR/owl2-overview/.

[2] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *International Symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer, 1999.

[3] Ian F Akyildiz and Ismail H Kasimoglu. Wireless sensor and actor networks: research challenges. *Ad hoc networks*, 2(4):351–367, 2004.

[4] Valerio F Annese, Giuseppe E Biccario, Silvia Cipriani, and Daniela De Venuto. Organoleptic properties remote sensing and life-time prediction along the perishables goods supply-chain. In *Proceedings of the 8th International Conference on Sensing Technology*, pages 130–135, 2014.

[5] Luigi Atzori, Antonio Iera, and Giacomo Morabito. From "smart objects" to "social objects": The next evolutionary step of the internet of things. *Communications Magazine, IEEE*, 52(1):97–105, 2014.

[6] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.

[7] Majid Bahrepour, Nirvana Meratnia, Mannes Poel, Zahra Taghikhaki, and Paul JM Havinga. Distributed event detection in wireless sensor networks for disaster management. In *Intelligent Networking and Collaborative Systems (INCOS), 2010 2nd International Conference on*, pages 507–512. IEEE, 2010.

[8] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. C-SPARQL: SPARQL for continuous querying. In *Proceedings of the 18th international conference on World wide web*, pages 1061–1062. ACM, 2009.

[9] Payam Barnaghi, Philippe Cousin, Pedro Malo, Martin Serrano, and Cesar Viho. Simpler IoT word(s) of tomorrow, more interoperability challenges to cope today. *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*, pages 277–329, 2013.

[10] Jenay Beer, Arthur D Fisk, and Wendy A Rogers. Toward a framework for levels of robot autonomy in human-robot interaction. *Journal of Human-Robot Interaction*, 3(2):74, 2014.

[11] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[12] Andre Bolles, Marco Grawunder, and Jonas Jacobi. Streaming SPARQL-extending SPARQL to process data streams. In *European Semantic Web Conference*, pages 448–462. Springer, 2008.

[13] Alexander Borgida. Description logics in data management. *IEEE transactions on knowledge and data engineering*, 7(5):671–682, 1995.

[14] Mike Botts, George Percivall, Carl Reed, and John Davidson. OGC® sensor web enablement: Overview and high level architecture. In *GeoSensor networks*, pages 175–190. Springer, 2008.

[15] Alan J Broder. Strategies for efficient incremental nearest neighbor search. *Pattern Recognition*, 23(1-2):171–178, 1990.

[16] Matt Calder, Robert A Morris, and Francesco Peri. Machine reasoning about anomalous sensor data. *Ecological Informatics*, 5(1):9–18, 2010.

[17] Kening Cao, Yongheng Wang, and Fengjuan Wang. Context-aware distributed complex event processing method for event cloud in internet of things. *Advances in Information Sciences & Service Sciences*, 5(8), 2013.

[18] Liming Chen, Chris D Nugent, and Hui Wang. A knowledge-driven approach to activity recognition in smart homes. *Knowledge and Data Engineering, IEEE Transactions on*, 24(6):961–974, 2012.

[19] Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M Donini, Azzurra Ragone, and Raffaele Rizzi. A semantic-based fully visual application for matchmaking and query refinement in b2c e-marketplaces. In *Proceedings of the 8th international conference on Electronic commerce*, pages 174–184. ACM, 2006.

[20] Michael Compton, Payam Barnaghi, Luis Bermudez, RaúL GarcíA-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, et al. The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:25–32, 2012.

[21] OMG CORBA and IIOP Specification. Object Management Group. *Joint revised submission OMG document orbos/99-02*, 1999.

[22] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[23] Li Da Xu, Wu He, and Shancang Li. Internet of Things in industries: a survey. *Industrial Informatics, IEEE Transactions on*, 10(4):2233–2243, 2014.

[24] Emanuele Della Valle, Stefano Ceri, Davide Francesco Barbieri, Daniele Braga, and Alessandro Campi. A first step towards stream reasoning. In *Future Internet Symposium*, pages 72–81. Springer, 2008.

[25] A K Dey, G D Abowd, and D Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-computer interaction*, 16(2):97–166, 2001.

[26] Francesco M Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics. *Principles of Knowledge representation*, 1:191–236, 1996.

[27] Mohamad Eid, Ramiro Liscano, and Abdulmotaleb El Saddik. A universal ontology for sensor networks data. In *2007 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pages 59–62. IEEE, 2007.

[28] Mica R Endsley. Level of automation effects on performance, situation awareness and workload in a dynamic control task. *Ergonomics*, 42(3):462–492, 1999.

[29] Dieter Fensel, Ian Horrocks, Frank Van Harmelen, Deborah McGuinness, and Peter F Patel-Schneider. OIL: Ontology infrastructure to enable the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[30] Jesus Arias Fisteus, Norberto Fernández García, Luis Sánchez Fernández, and Damaris Fuentes-Lorenzo. Ztreamy: A middleware for publishing semantic streams on the web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 25:16–23, 2014.

[31] Giancarlo Fortino, Antonio Guerrieri, Michelangelo Lacopo, Matteo Lucia, and Wilma Russo. An agent-based middleware for cooperating smart objects. In *Highlights on Practical Applications of Agents and Multi-Agent Systems*, pages 387–398. Springer, 2013.

[32] Johannes Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.

[33] Sahin Cem Geyik, Boleslaw K Szymanski, and Petros Zerfos. Robust dynamic service composition in sensor networks. *Services Computing, IEEE Transactions on*, 6(4):560–572, 2013.

[34] Michael A Goodrich, Timothy W McLain, Jeffrey D Anderson, Jisang Sun, and Jacob W Crandall. Managing autonomy in robot teams: observations from four experiments. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 25–32. ACM, 2007.

[35] Filippo Gramegna, Saverio Ieva, Giuseppe Loseto, and Agnese Pinto. Semantic-enhanced resource discovery for CoAP-based sensor networks. In *5th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI 2013)*, pages 228–233. IEEE, jun 2013.

[36] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[37] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

[38] Robert J Howlett and Lakhmi C Jain. *Radial basis function networks 1: recent developments in theory and applications*, volume 66. Springer Science & Business Media, 2001.

[39] Xiaolin Jia, Quanyuan Feng, Taihua Fan, and Quanshui Lei. RFID technology and its applications in Internet of Things (IoT). In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 1282–1285. IEEE, 2012.

[40] Karuna P Joshi, Yelena Yesha, and Tim Finin. Automating cloud services life cycle through semantic technologies. *Services Computing, IEEE Transactions on*, 7(1):109–122, 2014.

[41] Raja Jurdak, Cristina Videira Lopes, and Pierre Baldi. A framework for modeling sensor networks. In *Proceedings of the Building Software for Pervasive Computing Workshop at OOPSLA*, volume 4, pages 1–5, 2004.

[42] Krasimira Kapitanova, Sang H Son, and Kyoung-Don Kang. Using fuzzy logic for robust event detection in wireless sensor networks. *Ad Hoc Networks*, 10(4):709–722, 2012.

[43] Jeong-Hee Kim, Hoon Kwon, Do-Hyeun Kim, Ho-Young Kwak, and Sang-Joon Lee. Building a service-oriented ontology for wireless sensor networks. In *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, pages 649–654. IEEE, 2008.

[44] Sotiris B Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31:249–268, 2007.

[45] Thomas R Kurfess. *Robotics and automation handbook*. CRC press, 2005.

[46] Han Li and Guofei Jiang. Semantic message oriented middleware for publish/subscribe networks. *Defense and Security*, pages 124–133, 2004.

[47] M. Malaimalavathani and R. Gowri. A survey on semantic web service discovery. In *Information Communication and Embedded Systems (ICICES), 2013 International Conference on*, pages 222–225. IEEE, 2013.

[48] Sonia Ben Mokhtar, Davy Preuveneers, Nikolaos Georgantas, Valérie Issarny, and Yolande Berbers. EASY: Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support. *Journal of Systems and Software*, 81(5):785–808, 2008.

[49] Martin Mühlenbrock, Oliver Brdiczka, Dave Snowdon, and Jean-Luc Meunier. Learning to detect user activity and availability from a variety of sensor data. *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 13–22, 2004.

[50] Sreerama K Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, 2(4):345–389, 1998.

[51] Tuan Anh Nguyen, Andrea Raspitzu, and Marco Aiello. Ontology-based office activity recognition with applications for energy savings. *Journal of Ambient Intelligence and Humanized Computing*, 5(5):667–681, 2014.

[52] Young Park, Gyeong Mi Heo, and Romee Lee. Blogging for informal learning: Analyzing bloggers' perceptions using learning perspective. *Educational Technology & Society*, 14(2):149–160, 2011.

[53] Adam Pease. *Ontology: A practical guide*. Articulate Software Press, Angwin, CA 94508, 2011.

[54] Dennis Pfisterer, Kay Romer, Daniel Bimschas, Oliver Kleine, Richard Mietz, Cuong Truong, Henning Hasemann, Alexander Kroller, Max Pagel, Manfred Hauswirth, et al. Spitfire: toward a semantic web of things. *Communications Magazine, IEEE*, 49(11):40–48, 2011.

[55] Katharina Rasch, Fei Li, Sanjin Sehic, Rassul Ayani, and Schahram Dustdar. Context-driven personalized service discovery in pervasive environments. *World Wide Web*, 14(4):295–319, 2011.

[56] Frank Rosenblatt. Principles of neurodynamics. *Spartan Book*, 1962.

[57] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[58] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 3. Prentice hall Upper Saddle River, 2009.

[59] David J Russomanno, Cartik R Kothari, and Omoju A Thomas. Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. *IC-AI*, pages 637–643, 2005.

[60] Michele Ruta, Simona Colucci, Floriano Scioscia, Eugenio Di Sciascio, and Francesco M Donini. Finding commonalities in RFID semantic streams. *The 5th International Workshop on RFID Technology Concepts, Applications, Challenges (IWRT 2011), Procedia Computer Science*, 5:857–864, 2011.

[61] Michele Ruta, Eugenio Di Sciascio, and Floriano Scioscia. Concept abduction and contraction in semantic-based P2P environments. *Web Intelligence and Agent Systems*, 9(3):179–207, 2011.

[62] Michele Ruta, Floriano Scioscia, and Eugenio Di Sciascio. Enabling the semantic web of things: framework and architecture. *Sixth IEEE International Conference on Semantic Computing (ICSC 2012)*, pages 345–347, sep 2012. doi: 10.1109/ICSC.2012.42.

[63] Michele Ruta, Floriano Scioscia, and Eugenio Di Sciascio. A mobile matchmaker for resource discovery in the ubiquitous semantic web. In *Mobile Services (MS), 2015 IEEE International Conference on*, pages 336–343. IEEE, 2015.

[64] Michele Ruta, Floriano Scioscia, Giuseppe Loseto, and Eugenio Di Sciascio. Semantic-based resource discovery and orchestration in home and building automation: a multi-agent approach. *IEEE Transactions on Industrial Informatics*, 10(1):730–741, 2014.

[65] Tomasz Rybicki. Ontology recomposition. *Knowledge Engineering, Machine Learning and Lattice Computing with Applications*, pages 119–132, 2012.

[66] Mahadev Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal communications*, 8(4):10–17, 2001.

[67] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial intelligence*, 48(1):1–26, 1991.

[68] Schreiber, Guus and Raimond, Yves. RDF 1.1 Primer. https://www.w3.org/TR/rdf11-primer/, W3C Working Group Note 24 June 2014.

[69] Floriano Scioscia and Michele Ruta. Building a Semantic Web of Things: issues and perspectives in information compression. *Semantic Web Information Management (SWIM'09). In Proc. of the 3rd IEEE Int. Conf. on Semantic Computing (ICSC 2009)*, pages 589–594, 2009.

[70] Floriano Scioscia, Michele Ruta, Giuseppe Loseto, Filippo Gramegna, Saverio Ieva, Agnese Pinto, and Eugenio Di Sciascio. A mobile matchmaker for the ubiquitous semantic web. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(4):77–100, 2014.

[71] Nigel Shadbolt. Ambient intelligence. *IEEE intelligent Systems*, 18(4):2–3, 2003.

[72] Steffen Staab and Rudi Studer. *Handbook on ontologies*. Springer Science & Business Media, 2013.

[73] L Stein. Probability and the weighing of evidence. *British journal of social medicine*, 4(3):170–171, 1950.

[74] Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra. *Modular ontologies: concepts, theories and techniques for knowledge modularization*, volume 5445. Springer, 2009.

[75] Emanuele Della Valle, Stefano Ceri, Frank van Harmelen, and Dieter Fensel. It's a streaming world! reasoning upon rapidly changing information. *IEEE Intelligent Systems*, 24(6):83–89, 2009.

[76] Juan Ignacio Vazquez, Diego López De Ipiña, and Iñigo Sedano. Soam: an environment adaptation model for the pervasive semantic web. In *Computational Science and Its Applications-ICCSA 2006*, pages 108–117. Springer, 2006.

[77] W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview (Second Edition). http://www.w3.org/TR/owl2-overview/ 2016.09.27, W3C Recommendation 11 December 2012.

[78] W3C SPARQL Working Group. SPARQL 1.1 Overview. https://www.w3.org/TR/sparql11-overview/, Recommendation 21 March 2013.

[79] Mark Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.

[80] Rui Xu, Donald Wunsch, et al. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[81] Juan Ye, Stamatia Dasiopoulou, Graeme Stevenson, Georgios Meditskos, Efstratios Kontopoulos, Ioannis Kompatsiaris, and Simon Dobson. Semantic web technologies in pervasive computing: A survey and research roadmap. *Pervasive and Mobile Computing*, 23:1–25, 2015.

[82] Tong Zhang. An introduction to support vector machines and other kernel-based learning methods. *AI Magazine*, 22(2):103, 2001.

List of publications

Publications in proceedings of international conferences

1. Eliana Bove, Annarita Cinquepalmi, Danilo De Filippis, Filippo Gramegna, Saverio Ieva, Giuseppe Loseto, Agnese Pinto. *A semantic-based framework for RFID-assisted port supply chains.* In Toward Emerging Technology for Harbour sYstems and Services (TETHYS 2014 Workshop) – July 2014.

2. Agnese Pinto, Floriano Scioscia, Giuseppe Loseto, Michele Ruta, Eliana Bove, Eugenio Di Sciascio. *A semantic-based approach for Machine Learning data analysis.* In Ninth IEEE International Conference on Semantic Computing (ICSC 2015), page 324-327 – 2015.

3. Eliana Bove. *Object (b)logging: semantically rich context mining and annotation in pervasive environments.* In 6th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI 2015), page 210-215 – June 2015.

4. Floriano Scioscia, Saverio Ieva, Giuseppe Loseto, Eliana Bove, Danilo De Filippis. *ARGES: pAssengeRs and loGistics information Exchange System.* In Toward Emerging Technology for Harbour sYstems and Services (TETHYS 2015) – December 2015.

5. Michele Ruta, Floriano Scioscia, Eliana Bove, Annarita Cinquepalmi, Eugenio Di Sciascio. *A Knowledge-based Approach for Resource Discovery and Allotment in Swarm Middleware.* In SET-222 Specialists' Meeting on 'Swarm Centric Solution for Intelligent Sensor Networks' – June 2016.

6. Eliana Bove. *Semantic-Based Context Mining and Sharing in Smart Object Networks.* In The Tenth International Conference on Advances in Semantic Processing (SEMAPRO 2016) – October 2016.

7. Michele Ruta, Floriano Scioscia, Eliana Bove, Annarita Cinquepalmi, Eugenio Di Sciascio. *A Semantic-based Approach for Resource Discovery and Allocation in Distributed Middleware*, ACM/IFIP/USENIX Middleware 2016 – December 2016.

Publications in national workshops

1. Eugenio Di Sciascio, Daniela De Venuto, Simona Colucci, Tommaso Di Noia, Marina Mongiello, Michele Ruta, Giuseppe Loseto, Phuong Nguyen, Floriano Scioscia, Valerio F. Annese, Giorgio Basile, Giovanna Capurso, Silvia Cipriani, Danilo De Filippis, Filippo Gramegna, Saverio Ieva, Pasquale Pazienza, Lucrezia Rutigliani, Vincenzo Scarola, Eliana Bove, Annarita Cinquepalmi, Silvia Giannini, Hasan Ali Khattak, Vito C. Ostuni, Agnese Pinto, Jessica Rosati, Paolo Tomeo. *SisInfLab Research Group.* In 1st WORKSHOP on the State of the art and Challenges Of Research Efforts at Politecnico di Bari (SCORE 2014) – December 2014.

Academic activities

Summer schools

1. IOT360, *Internet of Things*. Rome, from 29-10-2014 to 01-11-2014. **First prize** at the IOT360 Hackathon event (part of IOT360 Summer School).

2. MLCI-2015, *Machine Learning*. Genoa, from 06-07-2015 to 10-07-2015.

3. RW2015, *Web Reasoning*. Berlin, from 31-07-2015 to 04-08-2015.

Seminars followed

1. *Swarm robotics research at iridia*, Prof. Marco Dorigo, Université Libre de Bruxelles, 8-04-2014.

2. *Mining user taste signals: combining recommender system*, Prof. Matthew Rowe, 10-11-2014

3. *Nuova programmazione H2020*, Dr. Alessio Gugliotta, Politecnico di Bari, 21-11-2014.

4. *Recommender Systems: an introduction*, Prof. Markus Zanker, Alpen Adria Universitat Klagenfurt, Austria, 25-11-2014.

5. *Apulian I-CiTies 2016, Laboratorio Nazionale CINI 'Smart Cities and Communities'*, Università degli Studi di Bari, 19-04-2015.

6. *IEEE training proposal*, Eszter Lukacs (IEEE training manager), 21-04-2015.

7. *Nuovi orizzonti per le Smart City*, Dr. Alessio Gugliotta, Politecnico di Bari, 12-06-2015.

8. *Challanges of vision - controlled micro flying robots: from frame-based to event-based vision*, Prof. Davide Scaramuzza, University of Zurich, 25-09-2015.

9. *Learning fuzzy descriptions from crisp OWL ontologies*, Prof. Umberto Straccia, ISTI-CNR Pisa, 14-10-2015.

10. *A gossip based distributed approach for heterogeneous multi-vehicle routing problems*, Dr. Mauro Franceschelli, Università di Cagliari, 20-11-2015.

11. *Robotic Seminar - affidabilità, elevate performance e facilità di programmazione della soluzione integrate di Mitsubishi Eletric*, Mitsubishi Electric, 10-12-2015.

12. *Introduction to Robotic Operating system (ROS)*, Dr. Donato Di Paola and Dr. Antonio Petitti, CNR, ISSIA, 22/29-06-2016 and 13/27-07-2016.

13. *Technologies & Innovation: developing business creation strategies*, Workshop, Politecnico di Bari, 21-11-2016.

Speaker at international conferences

1. *Toward Emerging Technology for Harbour sYstems and Services (TETHYS 2014 Workshop*, Bari, Italy, 15-07-2014.

2. *6th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI 2015 Workshop)*, Poster session, Gallipoli, Italy, 19-06-2015.

3. *Toward Emerging Technology for Harbour sYstems and Services (TETHYS 2015 Workshop)*, Bari, Italy, 14-12-2015.

4. *The Tenth International Conference on Advances in Semantic Processing (SEMAPRO 2016)*, Venice, Italy, 11-10-2016.

5. *ACM/IFIP/USENIX Middleware 2016*, Poster session, Trento, Italy, 14-12-2016.