Ubiquitous Systems and Ambient Intelligence

Software development based on components and services

M.I. Capel

ETS Ingenierías Informática y
Telecomunicación
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Granada
Email: manuelcapel@ugr.es
http://lsi.ugr.es/mcapel/

November, 20th 2020

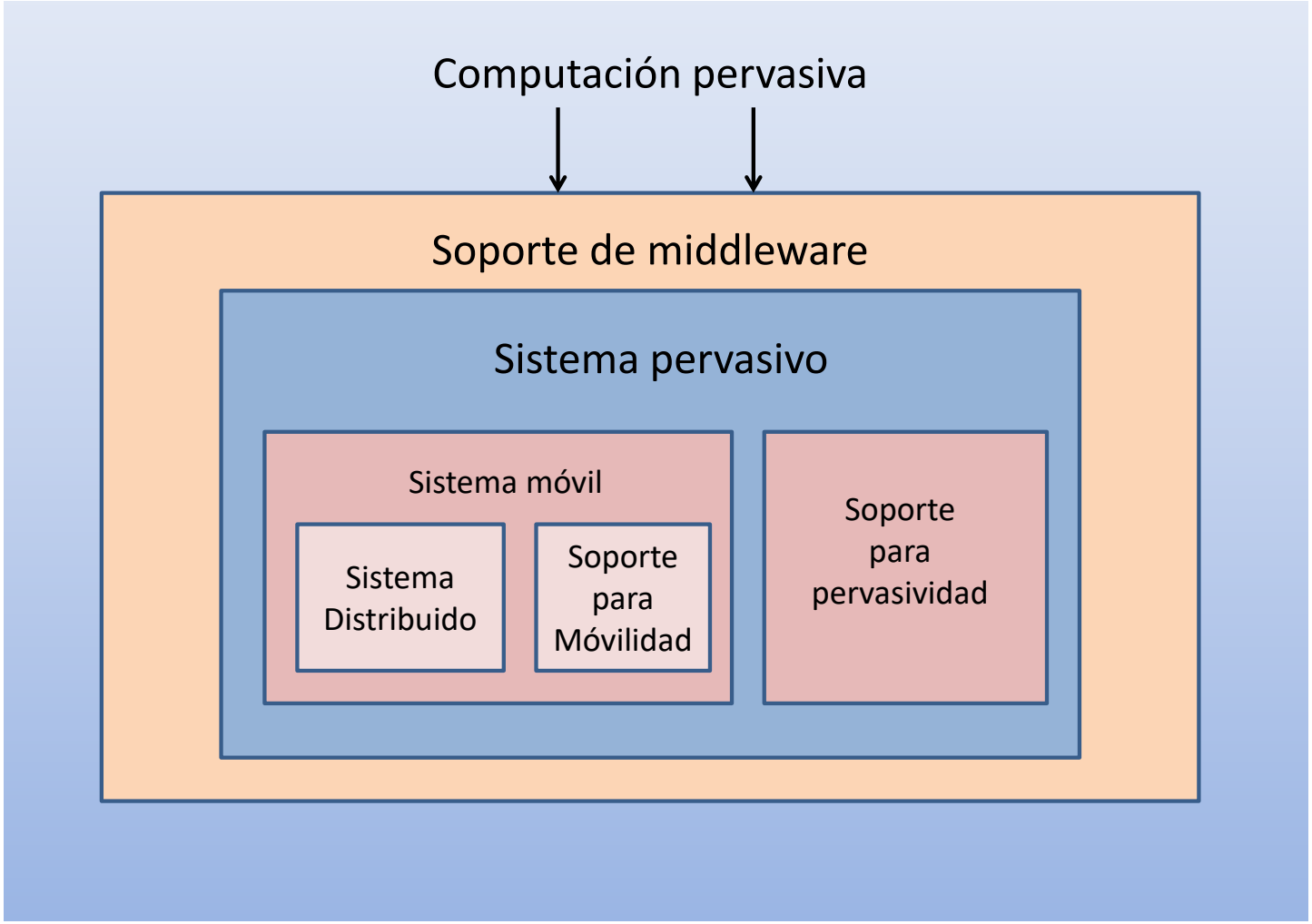
Máster Universitario en Ingeniería Informática

Introduction

History

- *Ubiquitous Computing* (UC) concept was introduced by visionaire Mark Weiser (1990):
  - Embedded, interconnected, intelligent systems that collaborate among them to carry out more and more complex tasks
- *Pervasivity*:
  - Growing tendency to incorporate computational capacity (usually in the form of microprocessors) in everyday objects to communicate . . . minimize user-computer interactions
  - “How information affects interactions with the built environments they occupy..." (Kecheng Liu , 2008)
- Conceptual exchange of concepts and terms: *Pervasive Computing*, *Ubiquitous Computation*, *Ambient Intelligence*, *Context Awareness*

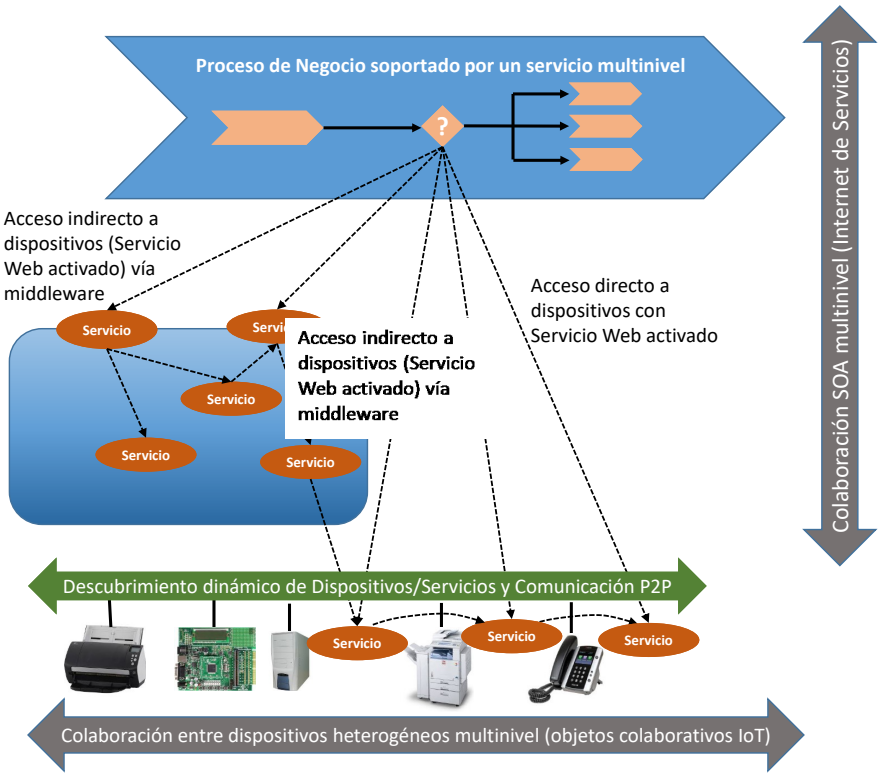
Technological support for pervasive computing



In

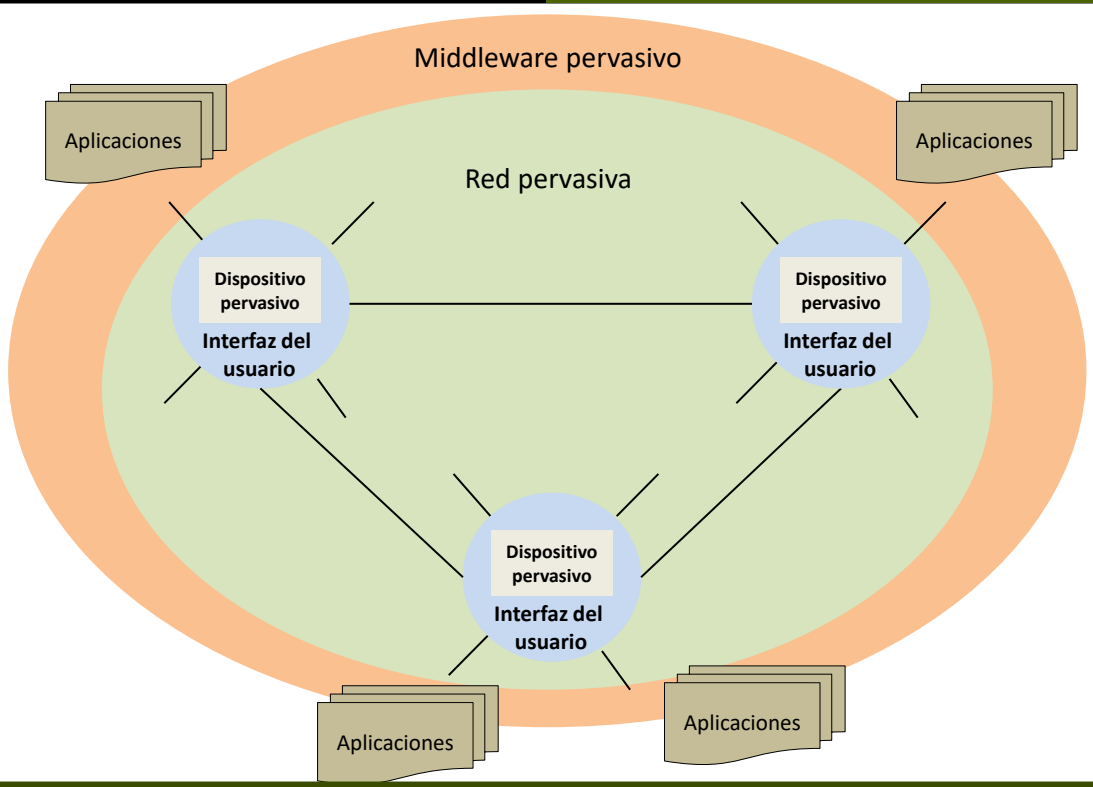
- The embedded devices with which we interact on a daily basis have got high communication and computation capabilities already, which have yield the following concepts:
  - “Internet of Things” [Atzori et al., 2010]
  - Comunic. “Machine-to-Machine”(M2M) [Kim et al., 2014]
- Ubiquitous Computing (UC) [Wang, 2004] propitiates the perfect convergence among the different technologies necessary to turn the “IoT” or “M2M” into reality
- There is not already an accepted standard for carrying out the development of ubiquitous systems [Stavropoulos et al., 2013]
- The high number of devices brings about “exponential” complexity, which is derived from the differences of software, networks and service composition

Multilevel SOA architectures for collaborative systems



The "software development" activity must be adapted to ubiquitous systems requirements

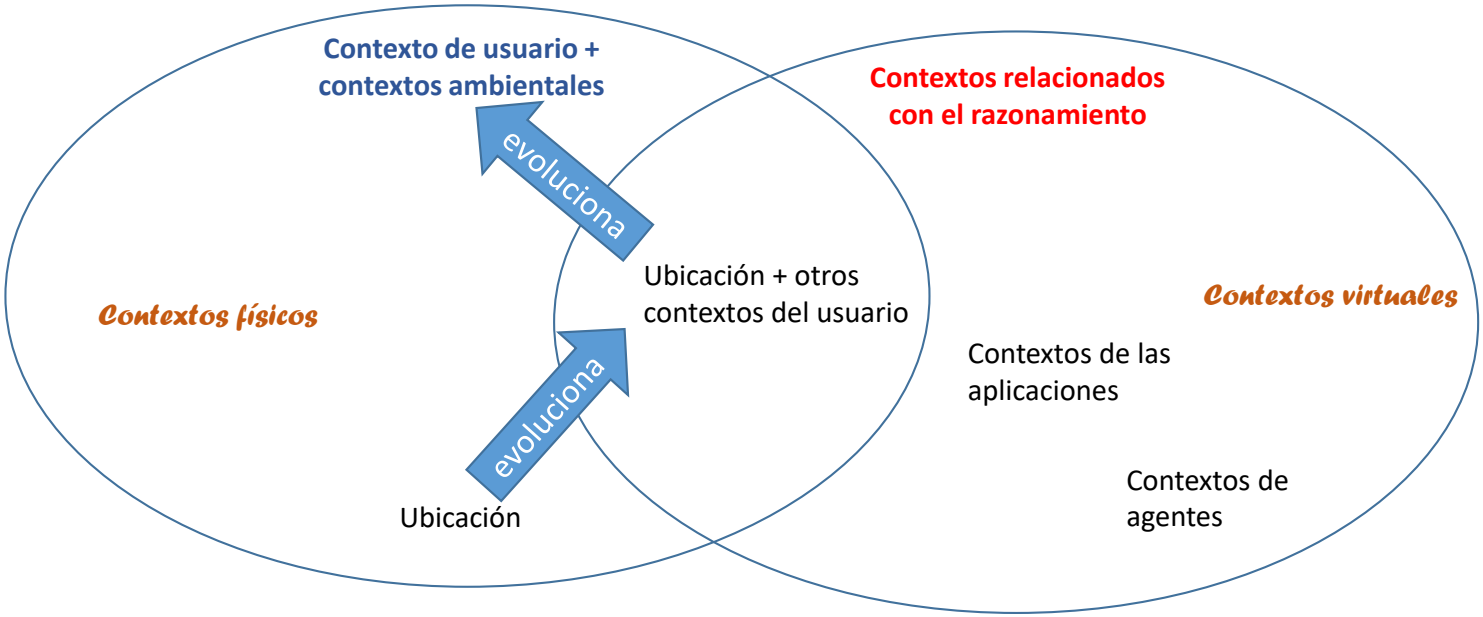
Solution: Through SOA, to make the computational-nodes functionality be interpreted as a service, which can be combined with others for more complex applications.



“Ubiquitous Space”

- Completely pervaded by devices
- “Transparent” functioning for its users
- “Colaborative” behavior
- Pro-activity of each device

Information, semantics and context



Context-aware computation

Its objective amounts to give a response to scenarios that originate in the application execution environment, depending of changing parameters and information detection that arises within a given context. This information, once obtained, is used for the system's interest that will be then able to react properly [Raz et al., 2006] [Madkour et al.,] to changing needs.

Information, semantics and context–II

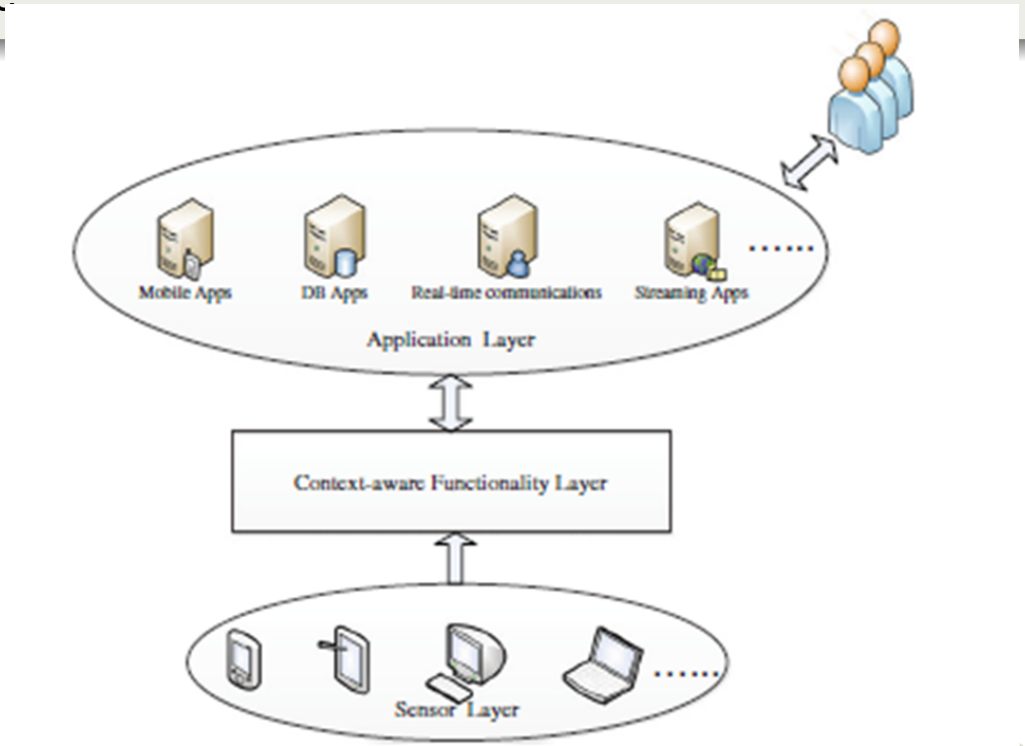
Context in Ubiquitous Computation environments

Set of

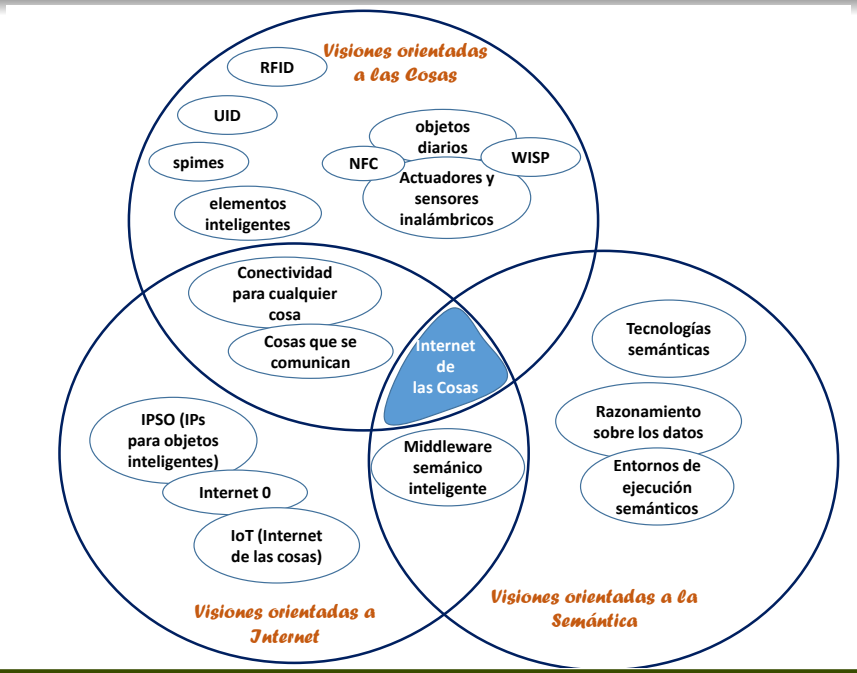
- properties that characterizes the execution environment of software applications,
- relationships between components and devices that fire events,
- actions that affect to individuals who do not have a direct role in the problem resolution but who can be affected by the solution of the problem

Importance of Context

- Propitiates the proactivity of system's services
- Reduces user direct participation in ubiquitous computing
- Increases the 'self–awareness" level of devices
- To have a "context model" is a fundamental element to obtain "ambient intelligence"



Importance of Context –II



Context Semantics

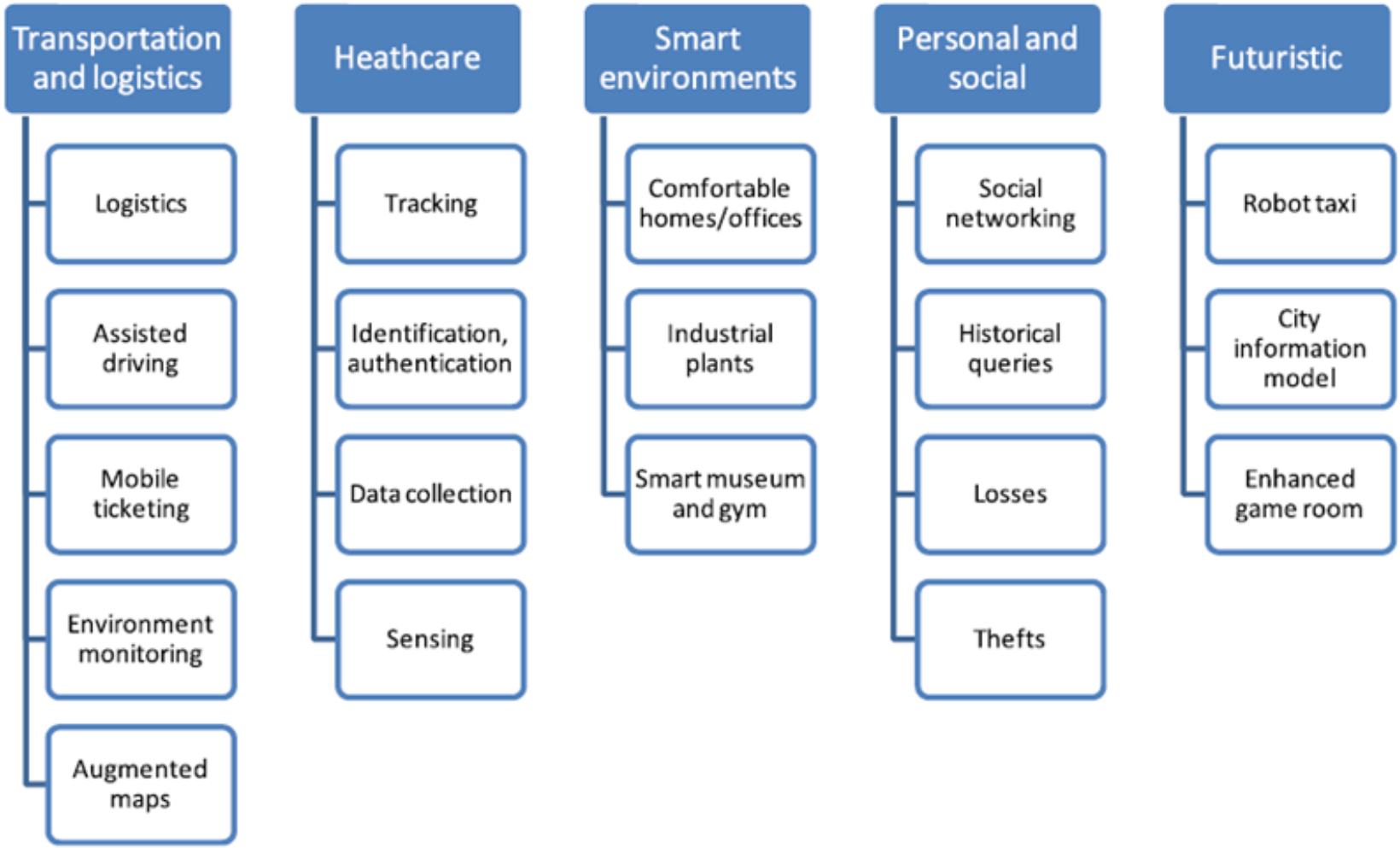
The development of *intelligent ambiances* is not only a software design problem, but it makes compulsory the definition of a domain of semantical information, within which devices and services will be deployed. It is necessary to establish a methodology that allow us to handle semantic information of systems and contexts to make possible the proactive behavior of system's services.

Traditional programming vs. ubiquitous environments

Fundamental ideas

- Improved adaptation to the execution context: "context–aware computation" [Lassila, 2005] [Perera et al., 2014]
- "Ambient Intelligence": independence wrt the system users
- Uniform representation of context, useful sensed information selection, conclusion extraction [Locke, 2006] [Baldauf et al., 2012]
- Limited number of computational nodes
- Standarisation of terms, context ontologies, automation

Domains and main application scenarios



Characteristics of a ubiquitous computation system

Essential elements

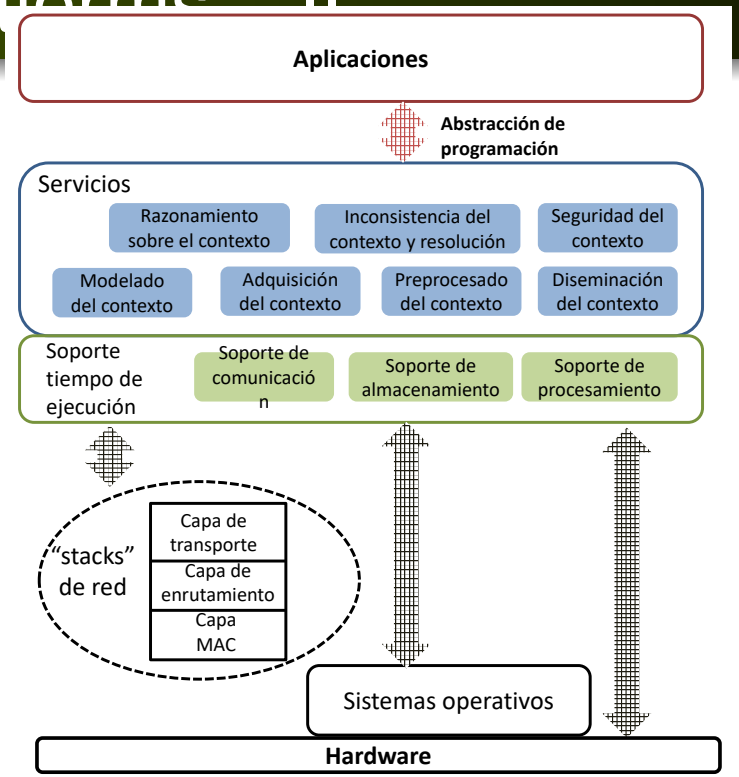
- “Invisible” interface
- “Open standards” deployment
- Use of low cost harware
- System’s reactivity and proactivity
- Execution locality of needed actions
- Environment adptation

Ubiquitous Platforms

Ubiquitous Computation [Weiser, 1993]

- Size reduction of hardware devices
- Lower enery consumption
- Increased execution velocity and memorya access
- “Device invisibility”
- Mobile computation sensor networks

Ubiquitous Plataforms II



Characteristics of a ubiquitous computation environment

- Dynamic reconfiguration capability
- modularity
- extensibility and portability

Ubiquitous Plataforms – III

New models

- Architectural styles: peer–to–peer, blackboard, etc.; centralized software architectures must be avoided
- Transparency, heterogeneity and interoperability of devices [Banavar and etal., 2000] [Guinard et al., 2010]
- Specific Middleware for service discovery, selection, adjustment and composition of services

Ubiquitous Plataforms – IV

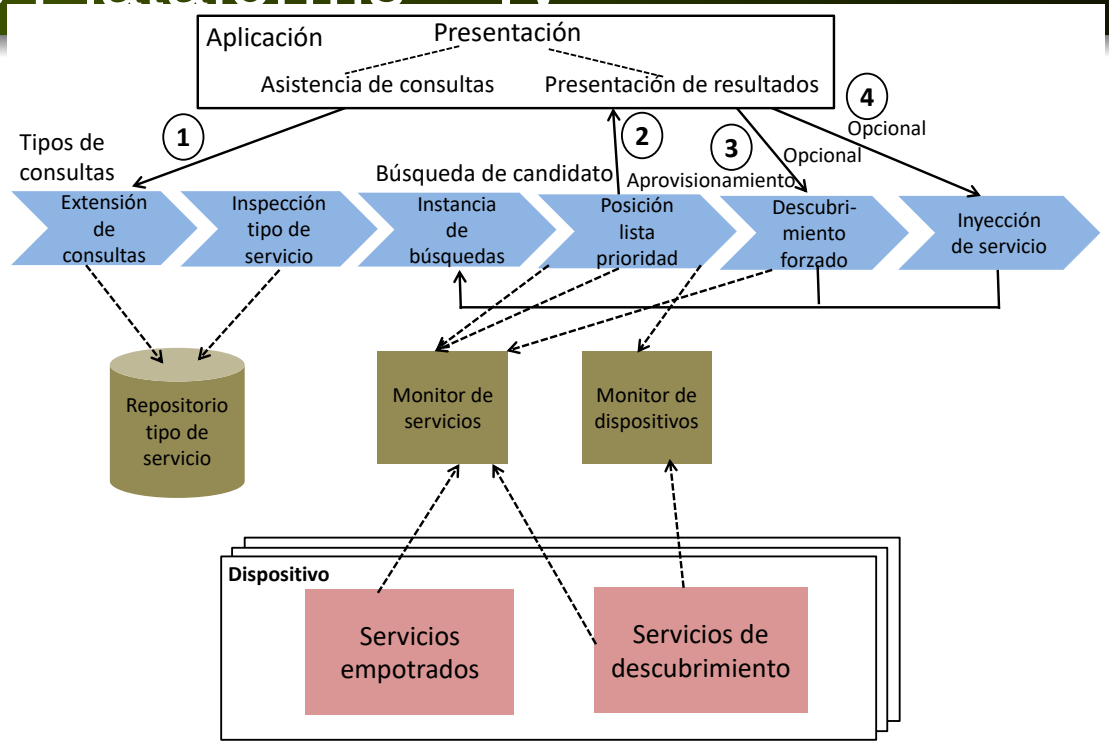
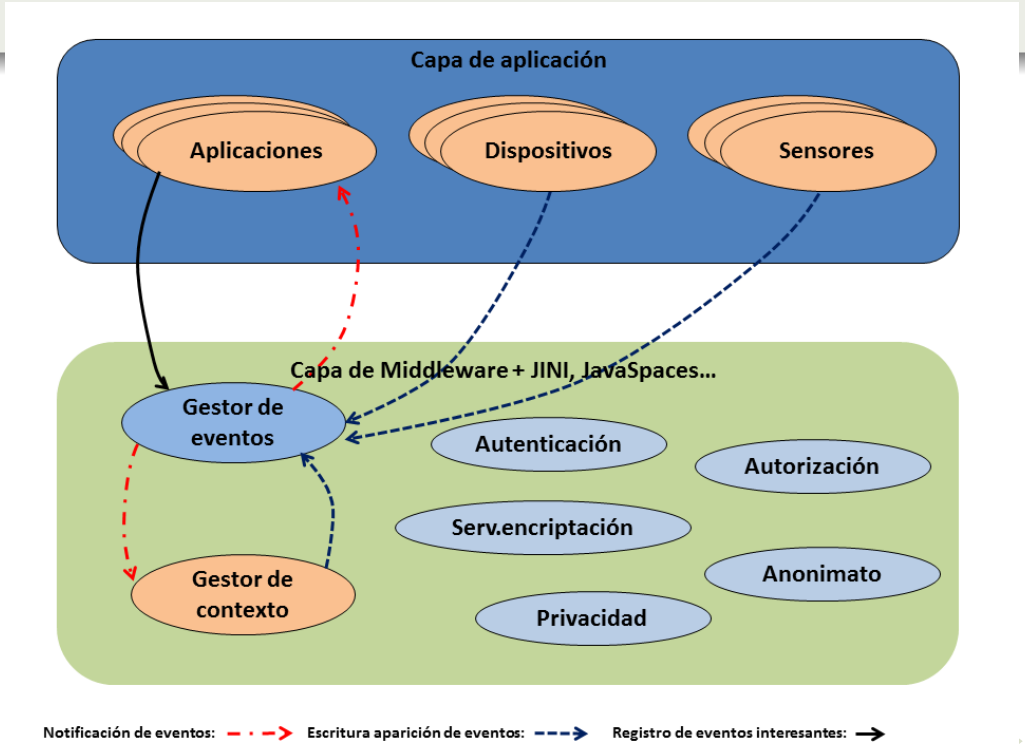


Figure: Service Discovery Process and Provisionning in Device Profile for WS(DPWS) [Guinard et al., 2010]

- SOCRADES Integration Architecture (SIA)
- SIA allows the ubiquitous integration of company services and real-world services that execute on embedded devices

Description of Java-based technologies

- Jini
- JavaSpaces
- Message Oriented Middleware (MOM)
- Web Services (DPWS)
- JXTA



Jini

Fundamental Characteristics

- Based on the “service federation” concept
- Client/Server software architecture [Apache, 2014]
- Services directory through Apache *LookUp Service*, which defines *proxies* for accessing
- Proxies implement Java interfaces to represent *service contracts*
- Flexible service-discovery but *un-robust* mechanism

Java Spaces

Fundamental idea

- This framework is based on global space of *tuples*, which are shared by vendors and clients, together with network resources and objects
- The virtual tuple-space is the appropriate support for the definition of SOA Directory Service
- It is also convenient for carrying out publishing and discovery of location-based services

Message Oriented Middleware

“Message Oriented Middleware”(MOM)

- Asynchronous (non-blocking) message-passing infrastructure as for developing a SOA-principles based framework
- It simplifies the most coordination between services
- Very low service coupling, it is mainly intended for implementing computational nodes in networks

Web Services

Web Services (DPWS)

- The most spread out proposal at moment for development of systems taht follow SOA principles
- Each application turns into a componet of a *Web Service* within the global network of services
- The SLA is done by using WSDL
- The WS is located through UDDI
- It is based on the SOAP protocol
- A lot of current research publications use WS for ubiquitous computation systems development

P2P-based frameworks

JXTA

- It allows to carry out publication tasks, discovery, collaboration and interaction between services [Oracle, 2014] according to SOA
- The computational nodes of a P2P architecture are used in a double-way: they swap client and servers roles
- JXTA needs less management work than centralized architectures as the Jini one
- It is the best for distributed systems with very limited resources(embedded systems)
- Publishing and queries indexation: supernodes, *Rendez-Vous* and *Relay-peers*, which are very efficient for SOAs

JXTA middleware for colaborative systems [?]

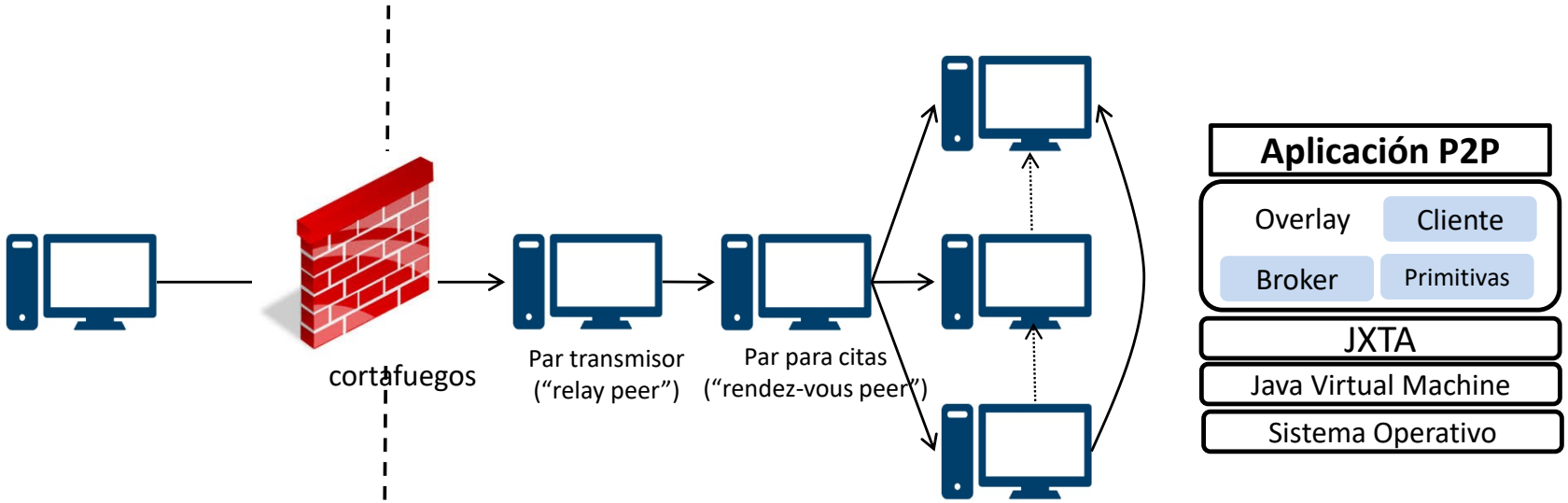


Figure: JXTA Middleware

JXTA middleware for colaborative systems-II

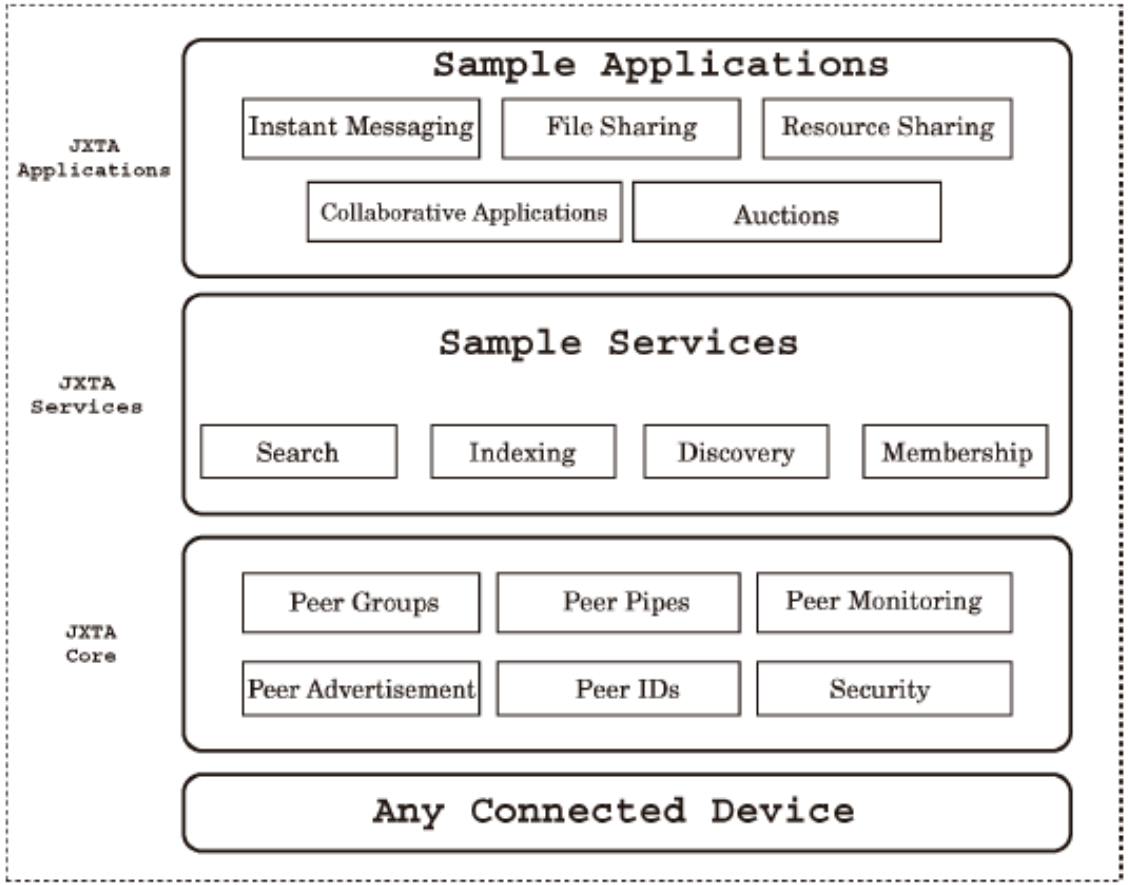


Figure: Layers and services that JXTA architecture offers

Summary of Java-based technologies

- Jini, JavaSpaces, MOM, Web Services and JXTA

	Jini	**JavaSpaces**	**MOM(JMS)**	**WS (DPWS)**	**JXTA**
Representation data	Java Marshal-ling Objects	Colections de Information	Messages	XML	XML
Transport	Protocols RMI-based	Serialization	Protocols RMI-based	SOAP over HTTP	TCP/IP or HTTP
Description Services	Interface of Java	Contract of Interface	Objects administered	WSDL	News and notices
Services Location	Service *Lookup*	Input points	Publish Subscribe	UDDI	Discovery services
Bindings languages	Java	Java	Java	Java,.NET Perl	Java,C
References remote	Objects *proxy*	Objects *proxy*	Identification	URL	Rendez-vous *peers*
Synchronicity	sync/async	-	async	sync	sync/async
Type of arquitecture	Client server	Client server	Client server	Client server	Peer-to peer

Colaborativity

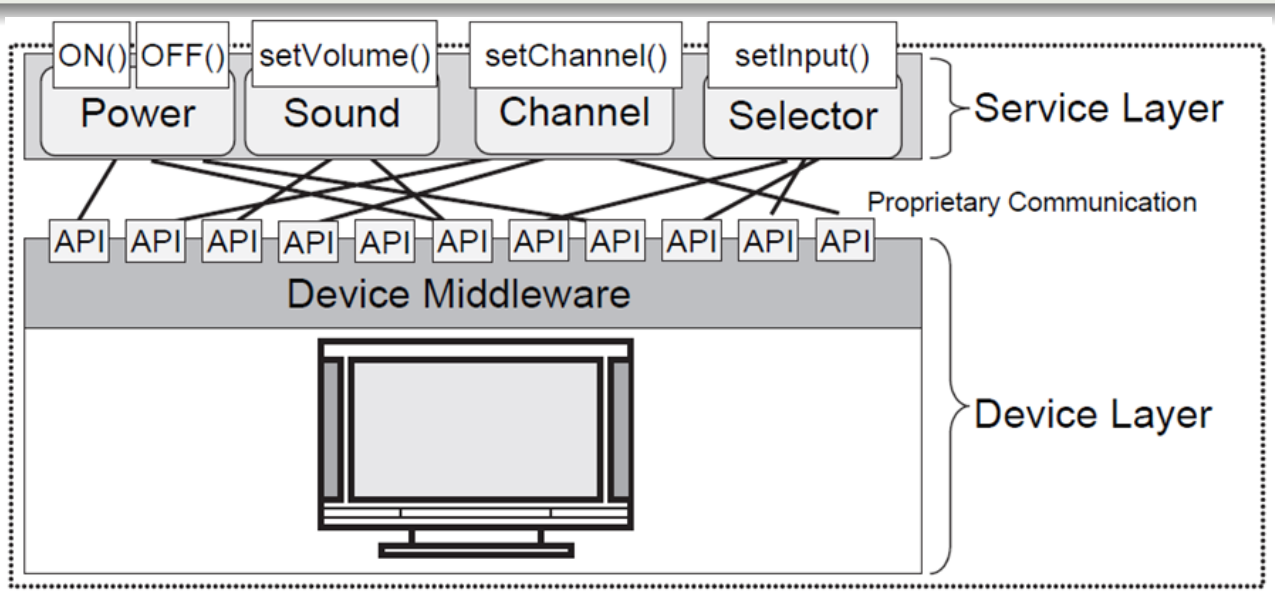
Fundamental ideas

- Proactivity of devices in ubiquitous spsces, propitiates AI-based systems
- Device in ubiquitous spaces: autnomous, distributed, interconnected, and colaborative
- Service according SOA architectures is a fundamental help for the development of ubiquitous applications
- Composition of services oriented to business modeling is not appropriate for Ubiquitous Computing
- Adaptation of SOA principles to specífic requirements of ubiquitous systems

Services composition

Fundamental ideas

- Objective of "*service composition*" in ubiquitous systems
- Ambiance Intelligence ⇔ Service Composition
- Service Composition ⇔ Collaborativeness



Services composition–II

Who (user) or what (device) is the responsible

When does the composition take place,

Where will the composition be carried out?

How does it get managed?

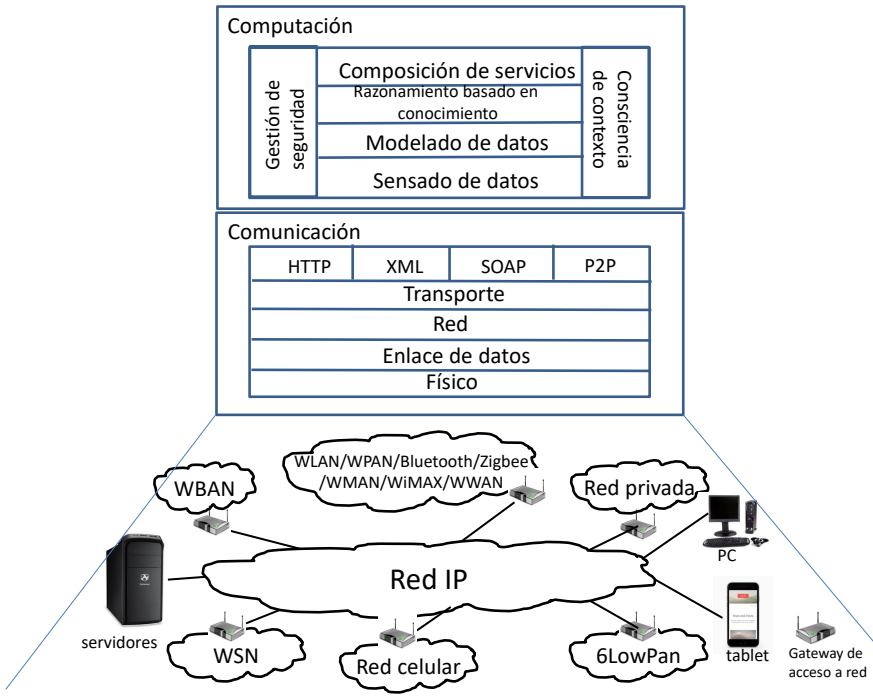


Figure: Ambiance intelligence system in an actual scenario, traversing networks and with several locations

Service Composition Conceptual Model

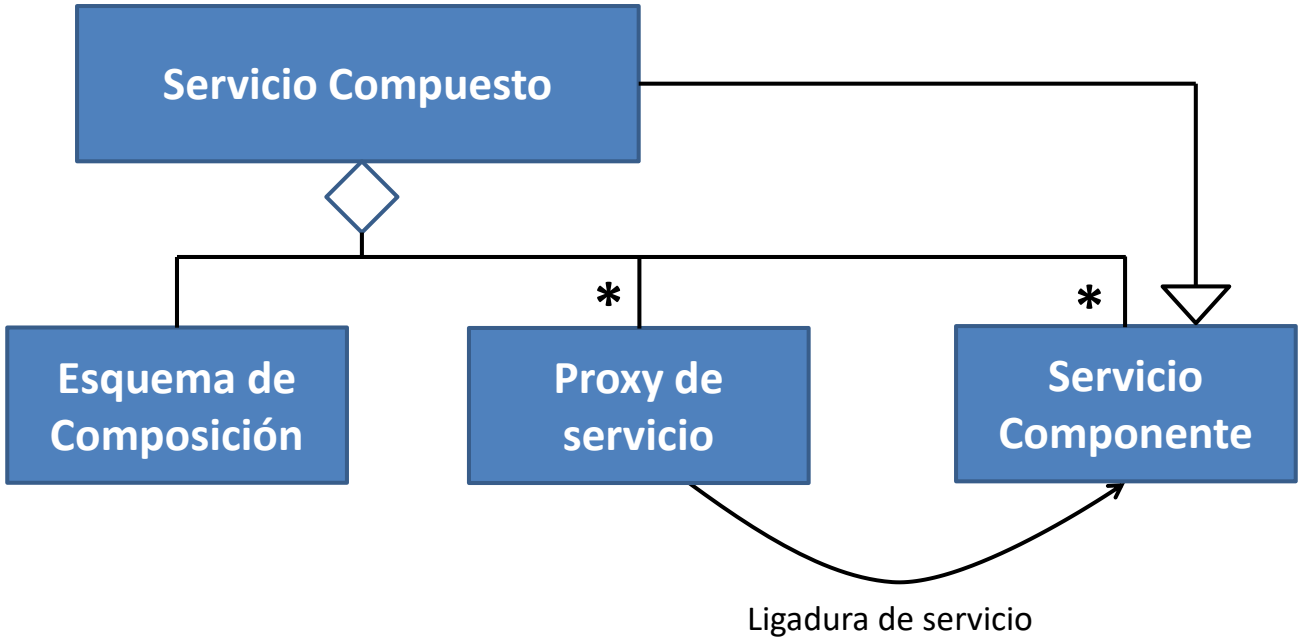


Figure: Conceptual model for ubiquitous systems

Services composition time

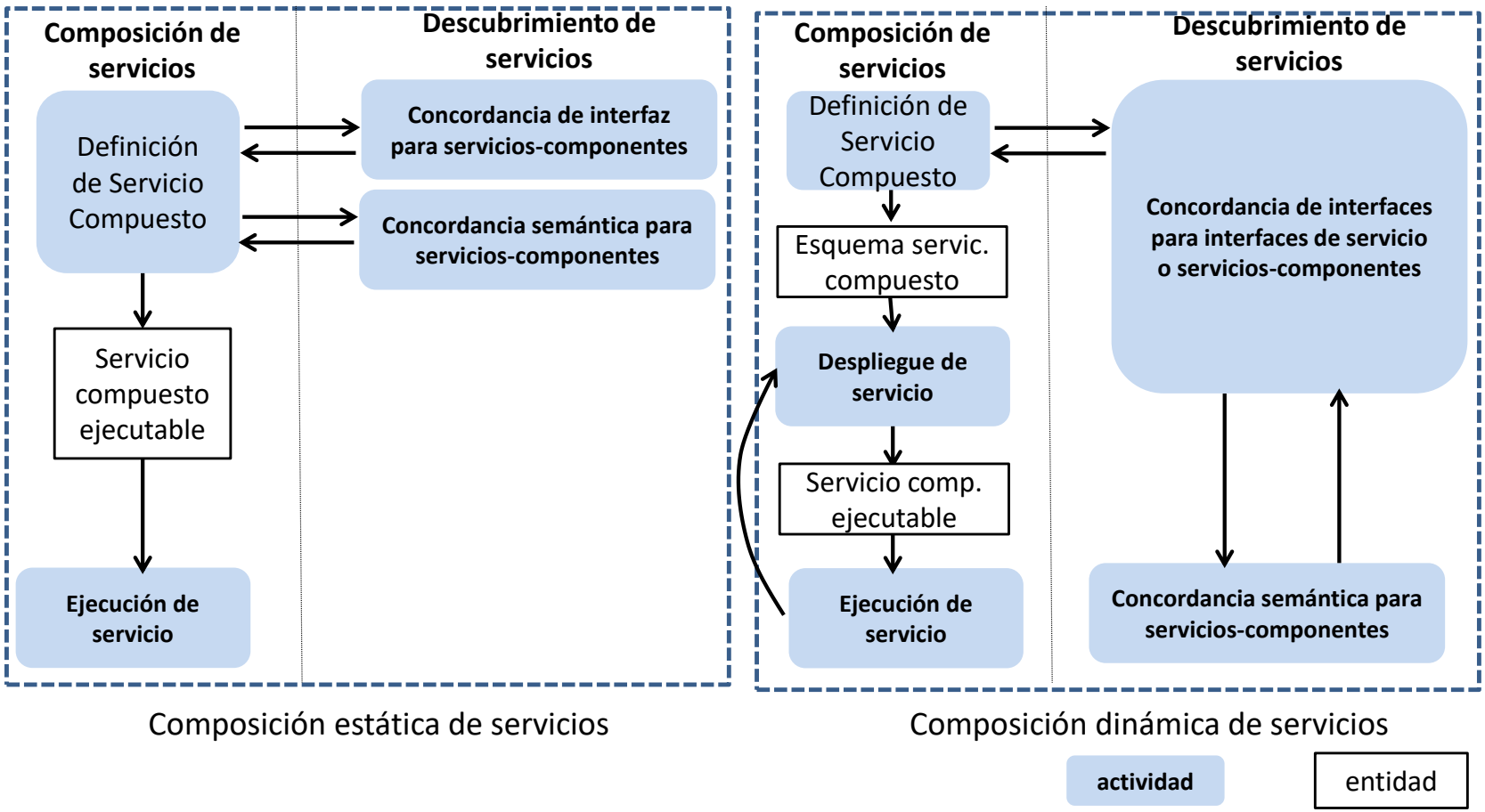


Figure: Types of services composition in ubiquitous systems

Service composition place

Centralized services composition [**?**]

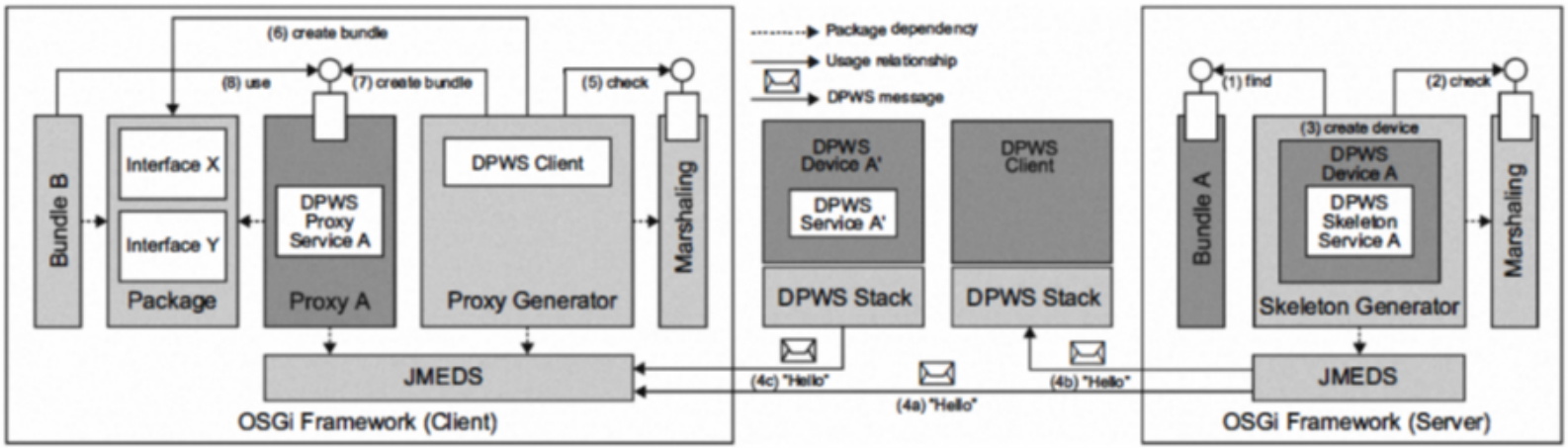


Figure: Proxies and skeletons composition in OSGi architecture

Open Services Gateway initiative(OSGi) architecture

Fundamental idea

- With OSGi we can implement a complete and dynamic model of software components, which does not exist in standalone Java/VM environments
- The components ("*bundles*") for deployment can be remotely installed, launched, stopped or updated without re-starting the systems
- OSGi, initially a specification for *gateways*, is currently used in Eclipse and plently of other IDEs, such as the ones for mobile-devices *apps*

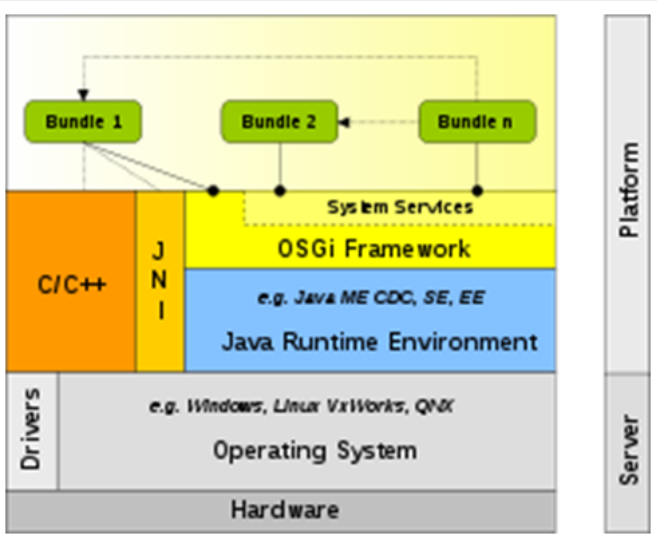


Figure: Layer-based design of OSGi system

Services composition validation

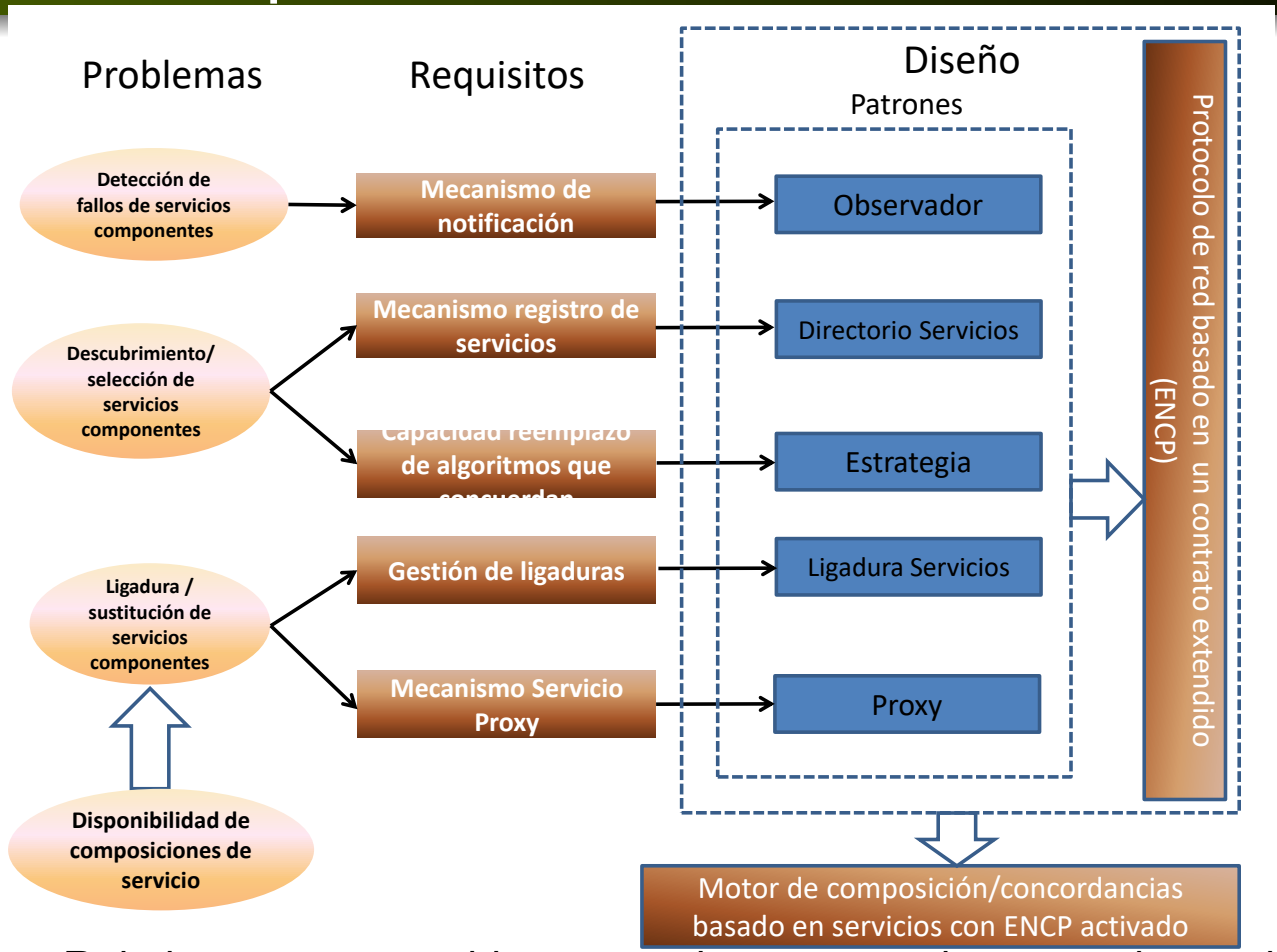


Figure: Relations among problems, requirements and patterns in services composition

Semantic information inclusion in services

Fundamental ideas

- Processable by devices directly, without human intervention
- Services are capable of finding devices and services with which to collaborate dynamically
- The necessity of having a notation to semantically describing entities and relationships within a given domain of significance

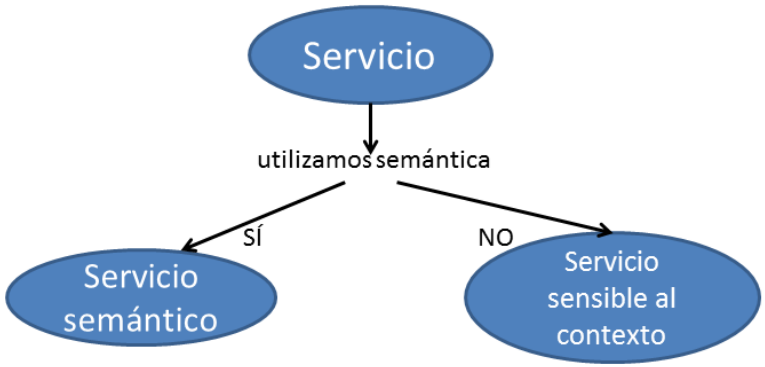


Figure: Semantics inclusion in Web services and applications

Semantic Web

- Notations for representation of concepts (almost all are defined in XML)
- XML is of not use for describing constraints that affect the meaning of structured documents
- RDF is data model for defining resources and relationships between them
- The ontology languages (RDF, OWL, OWL-S) allow for the definition of complex relationships between classes and characterization of properties

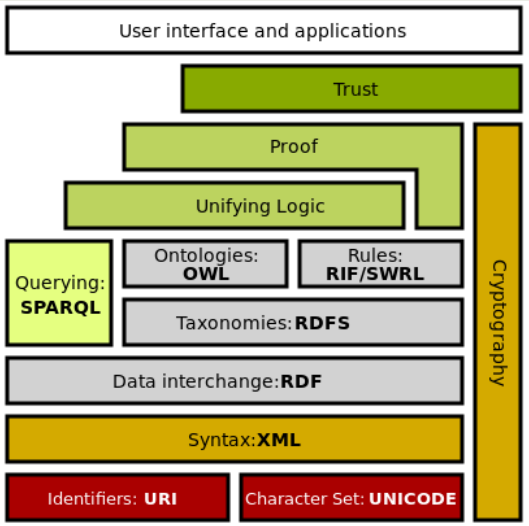


Figure: Semantic Web Structure by the W3C

RDF

Resource Description Framework (RDF)

- Considered as the *standard notation of the Semantic Web*
  - Standard model for data exchange between different domains
- Characteristics that propitiate data models fusion
- Evolution of schemes

"If the graph data model is the model the semantic web uses to store data, RDF is the format in which it is written".

RDF – II

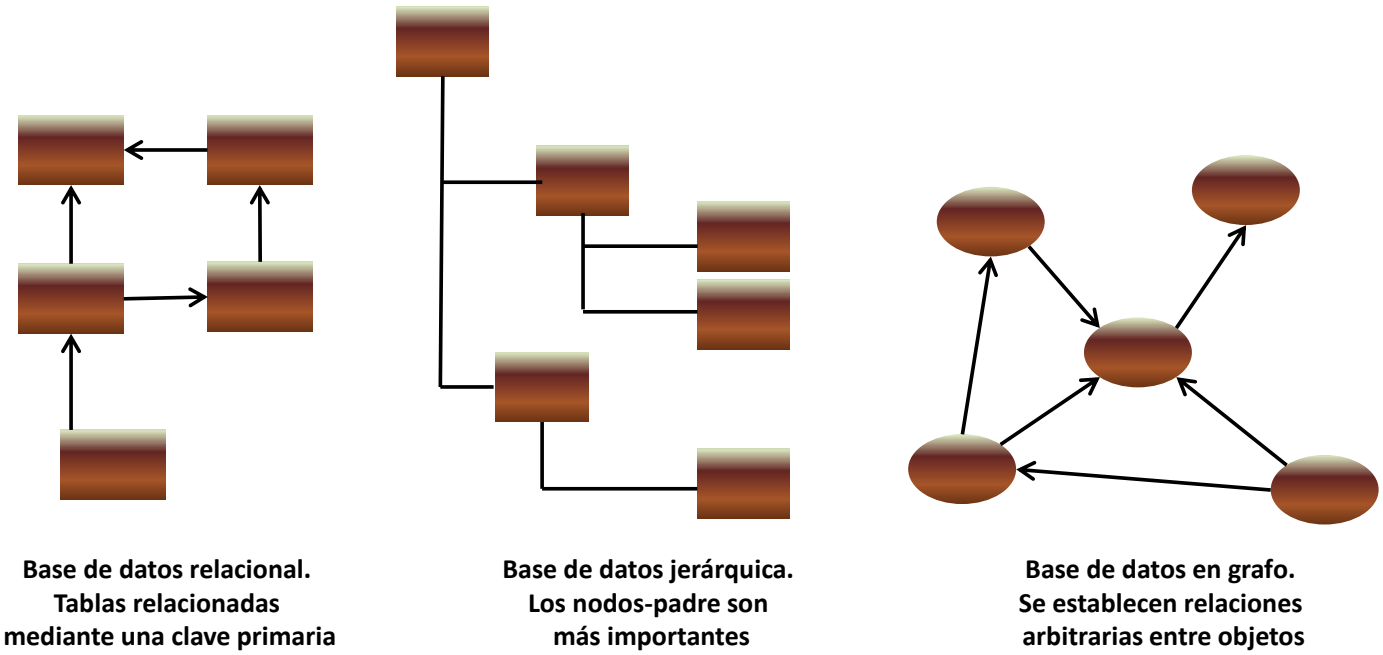
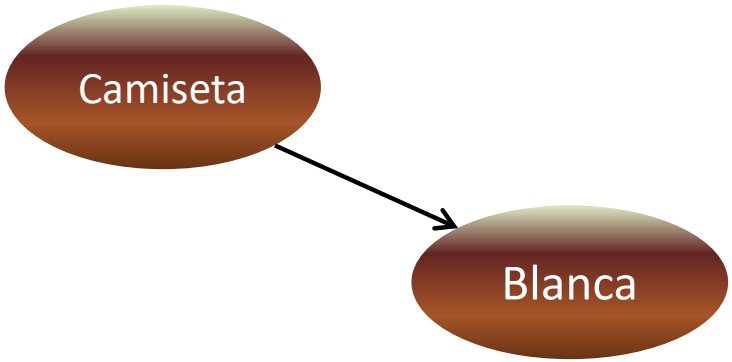


Figure: Data Base organized as a graph

Elements of RDF/XML

Triplets

- To give meaning to data
- Composition:
  - Subject
  - Predicate
  - Object



The RDF/XML statement

The RDF/XML in the following code (between the `<rdf:Description>` tags inclusive) is called an RDF statement, or sometimes called an RDF triple.

```
01.<?xml version="1.0" encoding="UTF-8"?>
02.<rdf:RDF
03.xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
04.xmlns:feature="http://www.linkeddatatools.com/clothing-features#">
05.
06.<rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">
07.<feature:size>12</feature:size> <!-- it is a literal value-->
08.<feature:color rdf:resource="http://www.linkeddatatools.com/colors#white"/>
09. <!--RDF objects can refer subjects of other statements-->
10.</rdf:Description>
11.</rdf:RDF>
```

Formalization of an RDF sentence

```
1.<rdf:Description rdf:about="sujeto">
2.<predicate rdf:resource="objeto" />
3.<predicate>valor literal</predicate>
4.<rdf:Description>
```

The `rdf:Description` RDF/XML element allows for grouping one or more statements into a single container. The above general form actually contains two statements referring to the same subject, but with two predicates and objects: a resource and a literal.

Exercise

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:region="http://www.country-regions.fake/">
<rdf:Description rdf:about="http://en.wikipedia.org/wiki/Granada
    ">
<dc:title>Granada</dc:title>
<dc:coverage>Andalucia</dc:coverage>
<dc:publisher>Wikipedia</dc:publisher>
<region:population>236,982</region:population>
<region:principaltown rdf:resource="http://www.country-regions.
    fake/granada"/>
</rdf:Description>
</rdf:RDF>
```

Identify on the RDF document above the: *subject* of the statement, *predicates of the statement* and including whether the objects referenced by the *predicates* are *resources* or *literals*

Solution to the exercise

Triples of the Data Model

Number	Subject	Predicate	Object
1	http://en.wikipedia.org/wiki/Granada	http://purl.org/dc/elements/1.1/title	"Granada"
2	http://en.wikipedia.org/wiki/Granada	http://purl.org/dc/elements/1.1/coverage	"Andalucia"
3	http://en.wikipedia.org/wiki/Granada	http://purl.org/dc/elements/1.1/publisher	"Wikipedia"
4	http://en.wikipedia.org/wiki/Granada	http://www.country-regions.fake/population	"236,982"
5	http://en.wikipedia.org/wiki/Granada	http://www.country-regions.fake/principaltown	http://www.country-regions.fake/granada

To be completed...

Figure: *Triples* of the model, obtained with the validator W3C of RDF

XML namespace: URIs

XML namespace URIs in RDF are used to distinguish between properties with the same (“tag”) name. To get the fully qualified URI, simply substitute the namespace prefix with the namespace URI.

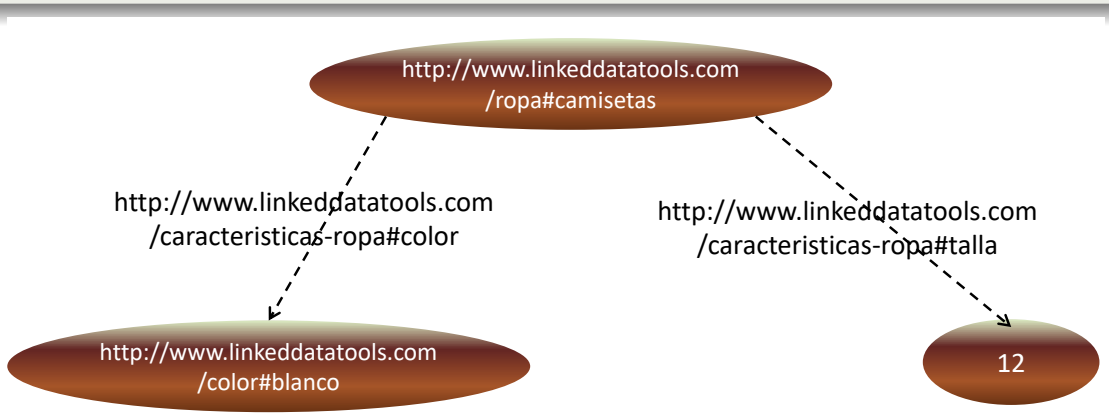


Figure: To obtain the qualified URI by its prefix substitution.

If we define the semantic space:

```
xmlns:feature="http://www.linkeddatatools.com/clothes-characteristics#"
```

Then graph label only needs to be, `characteristic:color`

Semantic modeling

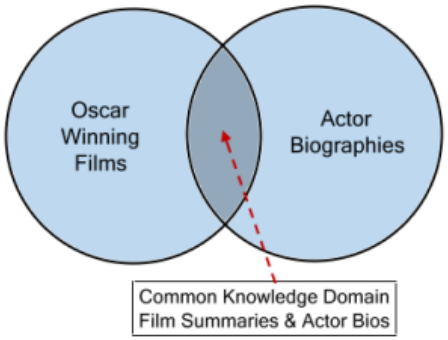
Fundamental ideas on RDF

- Model for data recording; these data can globally exchanged (from any remote location to another)
- does not plan anything for keeping meaning of stored data

Model	Format	Data	Metadata	Identifi-cation	Queries	Seman-tics
Serial. objects	.NET CLR objects	Values property	Values property	Name archive	LINK	N.D.
Relational	Oracle MySQL	Values cells	Definition cols. and rows	Values unique key	SQL	N.D.
Hierarchical	XML	Values tag/attrib.	XSD/ DTD	Values key/attrib.	XPath	N.D.
Graph	RDF/XML Turtle	RDF	RDFS/ OWL	URI	SPARQL	yes, with OWL

Knowledge Integration

Reason for including semantics associated to data.



This sort of information exchange across incompatible, independently designed, data systems takes time, money and human contextual interpretation of the different datasets. It is also restrictive to the data domains of these two websites only, any further additions to their knowledge from elsewhere will demand similar efforts. It requires humans to understand the meaning of the data and agree on common formats to make collaborate the two databases appropriately

Semantic modeling of data

Base Ontology + end–point that can be publicly queried

- The two sites (of both DB) can now query each other using the same terms.
- Later extension of the queried information
- To make easier carry out most complete queries
- This happens without the need for transformation, mapping, or contracts being set up between the two sites. It all happens through semantics.

International innitiatives – metadata

- Dublin Core Metadata Initiative (DCMI)
- Friend of A Friend (FOAF)
- OpenCyc

Ontology languages

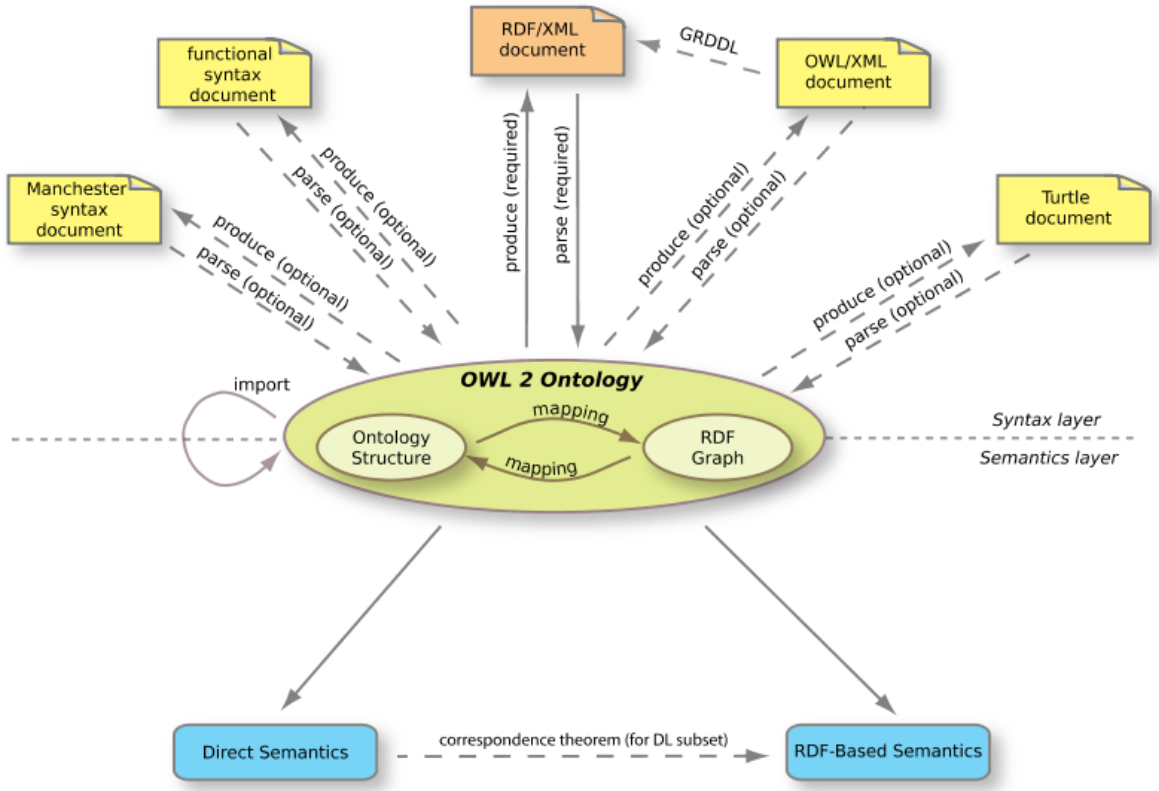


Figure: OWL 2.0 structure

Preliminar example with OWL

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
<!-- OWL Header example -->
<owl:Ontology rdf:about="http://www.linkeddatatools.com/plants">
<dc:title>The LinkedDataTools.com example of an ontology of
    plants</dc:title>
<dc:description>An example of ontology written for
    LinkedDataTools.com with RDFS and OWL </dc:description>
</owl:Ontology>
<!-- OWL Class definition example -->
<owl:Class rdf:about="http://www.linkeddatatools.com/plants#
    planttype">
<rdfs:label>The type of plant</rdfs:label>
<rdfs:comment>The class of plants types.</rdfs:comment>
</owl:Class>
</rdf:RDF>
```

Classes, subclasses and *individuals* in OWL

```
<!-- OWL subclass definition - Flower -->
<owl:Class rdf:about="http://www.linkeddatatools.com/plants#flowers">
<!-- Flowers is a subclass of planttype -->
<rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/plants#planttype"/>
<rdfs:label>Flowering plants</rdfs:label>
<rdfs:comment>Flowering plants, also known as angiosperms.</rdfs:comment>
</owl:Class>
<!-- OWL subclass definition - Shrub -->
<owl:Class rdf:about="http://www.linkeddatatools.com/plants#shrubs">
<!-- A shrub is a subclass planttype -->
<rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/plants#planttype"/>
<rdfs:label>Shrubs</rdfs:label>
<rdfs:comment>Shrubs, a type of plant that have branches from the base.</rdfs:comment>
</owl:Class>    ...
```

Parsing with the RDF validator

Triples of the Data Model

triples

Number	Subject	Predicate	Object
1	http://www.linkeddatatools.com/plants	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Ontology
2	http://www.linkeddatatools.com/plants	http://purl.org/dc/elements/1.1/title	"The LinkedDataTools.com example of an ontology of plants"
3	http://www.linkeddatatools.com/plants	http://purl.org/dc/elements/1.1/description	"An example of ontology written for LinkedDataTools.com with RDFS and OWL"
4	http://www.linkeddatatools.com/plants#planttype	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
5	http://www.linkeddatatools.com/plants#planttype	http://www.w3.org/2000/01/rdf-schema#label	"The type of plant"
6	http://www.linkeddatatools.com/plants#planttype	http://www.w3.org/2000/01/rdf-schema#comment	"The class of plants types."
7	http://www.linkeddatatools.com/plants#flowers	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
8	http://www.linkeddatatools.com/plants#flowers	http://www.w3.org/2000/01/rdf-schema#subClassOf	http://www.linkeddatatools.com/plants#planttype
9	http://www.linkeddatatools.com/plants#flowers	http://www.w3.org/2000/01/rdf-schema#label	"Flowering plants"
10	http://www.linkeddatatools.com/plants#flowers	http://www.w3.org/2000/01/rdf-schema#comment	"Flowering plants, also known as angiosperms."
11	http://www.linkeddatatools.com/plants#shrubs	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
12	http://www.linkeddatatools.com/plants#shrubs	http://www.w3.org/2000/01/rdf-schema#subClassOf	http://www.linkeddatatools.com/plants#planttype
13	http://www.linkeddatatools.com/plants#shrubs	http://www.w3.org/2000/01/rdf-schema#label	"Shrubs"
14	http://www.linkeddatatools.com/plants#shrubs	http://www.w3.org/2000/01/rdf-schema#comment	"Shrubs, a type of plant that have branches from the base."

Figure: *Triples* of the subclasses and individuals preliminar OWL example

Property definition with OWL

Properties

- Individuals in OWL are related by properties. There are two types of property in OWL:
  - Properties of ADTs (owl:DatatypeProperty): relates individuals (instances) of OWL classes to literal values
  - between objects(owl:ObjectProperty): relates individuals (*instances*) of two OWL classes

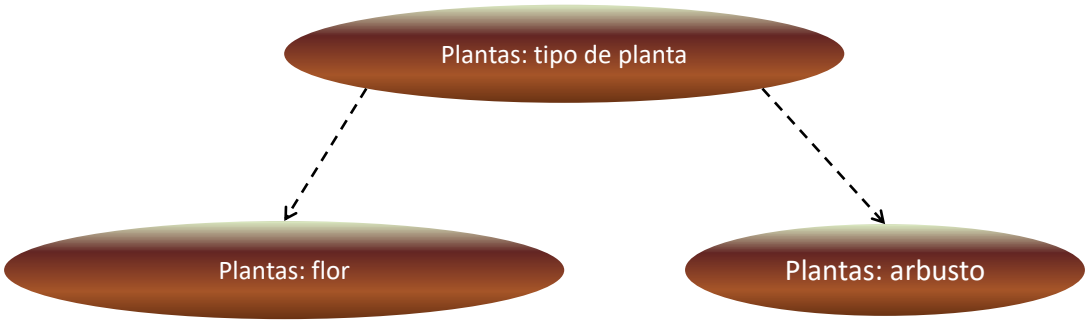


Figure: Term hierarchies (taxonomies) in OWL

Example of *data type* property expressed with OWL

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:plants="http://www.linkeddatatools.com/plants#">
<!-- OWL Header Omitted For Brevity -->
<!-- OWL Classes Omitted For Brevity -->
<!-- Define the family property -->
<owl:DatatypeProperty rdf:about="http://www.linkeddatatools.com/
    plants#family"/>
<rdf:Description rdf:about="http://www.linkeddatatools.com/
    plants#magnolia">
<!-- Magnolia is a type (instance) of the flowers class -->
<rdf:type rdf:resource="http://www.linkeddatatools.com/plants#
    flowers"/>
<!-- The magnolia is part of the 'Magnoliaceae' family -->
<plants:family>Magnoliaceae</plants:family>
</rdf:Description>
</rdf:RDF>
```

Parsing with the RDF validator-II

Triples of the Data Model

Number	Subject	Predicate	Object
1	http://www.linkeddatatools.com/plants#family	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty
2	http://www.linkeddatatools.com/plants#magnolia	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.linkeddatatools.com/plants#flowers
3	http://www.linkeddatatools.com/plants#magnolia	http://www.linkeddatatools.com/plants#family	"Magnoliaceae"

Figure: *Triples* of the datatype property example in OWL

Classes, subclasses and *individuals* in OWL–II

Fundamental idea

In OWL, properties are only understood with respect to their instances.
In the previous example, you may use the same 'family' property for an instance of a completely different class (.e.: for 'people' instead of 'plants').

Differences between POO–classes and OWL–class type

- The properties are owned by instances (*individuals*) and are not described in its class data type definition
- Consequences in the observable definition of instances (property independence)
- Property definition time

Example of an object property -one which links an instance to another instance-

```
1  <owl:DatatypeProperty rdf:about="http://www.linkeddatatools.com/
       plants#family"/>
2
3  <owl:ObjectProperty rdf:about="http://www.linkeddatatools.com/
       plants#similarlyPopularTo"/>
4
5  <!-- Orchid is an individual (instance) of the flowers class -->
6  <rdf:Description>
7  <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#
       flowers"/>
8  <!-- The orchid is part of the 'Orchidaceae' family -->
9  <plants:family>Orchidaceae</plants:family>
10
11 <!-- The orchid is similarly popular to the magnolia -->
12 <plants:similarlyPopularTo rdf:resource="http://www.
       linkeddatatools.com/plants#magnolia"/>
13 </rdf:Description>
```

Example of an object property -one which links an instance to another instance- II

```
<!-- Magnolia is an individual (instance) of the flowers class
     -->
<rdf:Description>
<rdf:type rdf:resource="http://www.linkeddatatools.com/plants#
    flowers"/>
<!-- The magnolia is part of the 'Magnoliaceae' family -->
<plants:family>Magnoliaceae</plants:family>

<!-- The magnolia is similarly popular to the orchid -->
<plants:similarlyPopularTo rdf:resource="http://www.
    linkeddatatools.com/plants#orchid"/>
</rdf:Description>
```

Parsing with the RDF validator-IV

Triples of the Data Model

Number	Subject	Predicate	Object
1	http://www.linkeddatatools.com/plants#family	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty
2	http://www.linkeddatatools.com/plants#similarlyPopularTo	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
3	genid:A13252	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.linkeddatatools.com/plants#flowers
4	genid:A13252	http://www.linkeddatatools.com/plants#family	"Orchidaceae"
5	genid:A13252	http://www.linkeddatatools.com/plants#similarlyPopularTo	http://www.linkeddatatools.com/plants#magnolia
6	genid:A13253	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.linkeddatatools.com/plants#flowers
7	genid:A13253	http://www.linkeddatatools.com/plants#family	"Magnoliaceae"
8	genid:A13253	http://www.linkeddatatools.com/plants#similarlyPopularTo	http://www.linkeddatatools.com/plants#orchid

Figure: *Triples* of the object property example in OWL

Exercise

Draw a graph showing the Orchid and Magnolia class instances and their predicates, according to the RDF graph defined above.

Solution to the exercise

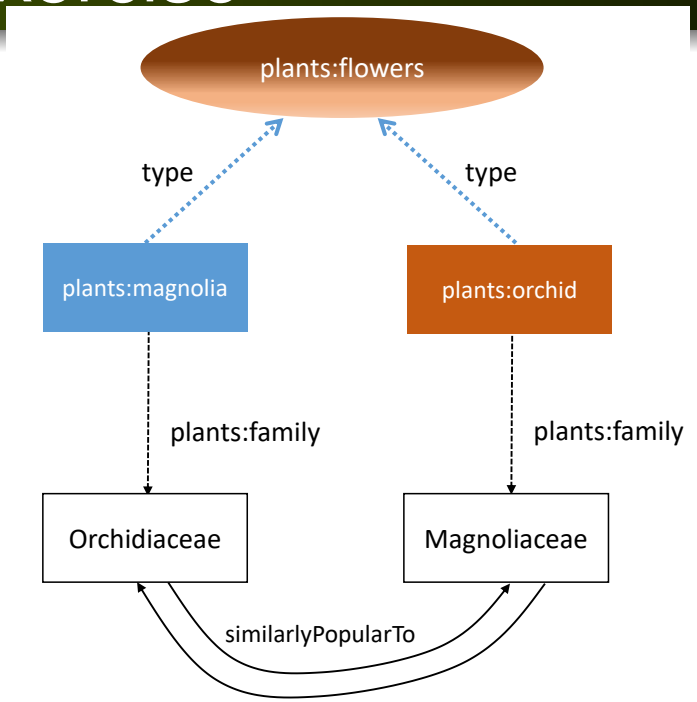


Figure: *similarlyPopularTo*: two-way relation between 2 instances of the a class OW

Note that:

- Object properties need not to be bi-directional - they may be one way
- Neither need they be between instances of the same OWL class: you can set them between completely different OWL classes

Web Ontology Language for Services (OWL-S)

- OWL-S is an ontology with the basic elements for describing services' capabilities from a semantical point of view
- OWL-S allows services's representation that own properties given by a predefined knowledge base
- Atomic services and compossed services in OWL-S
- Its main objective is to make possible automatic discovery, invokation and composition of services

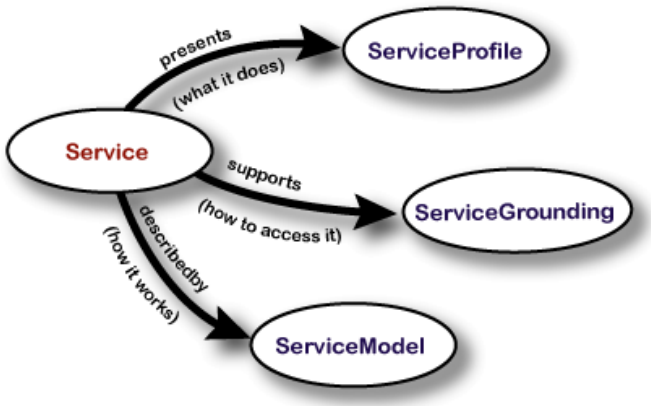


Figure: Upper part of an OWL-S ontology of services

Web Ontology Language for Services (OWL-S)–ii

Fundamental concepts regarding a service description

- Service Profile: what the service does?
- Service Model: how is the service used?
- Service Grounding: how can we interact with the service?

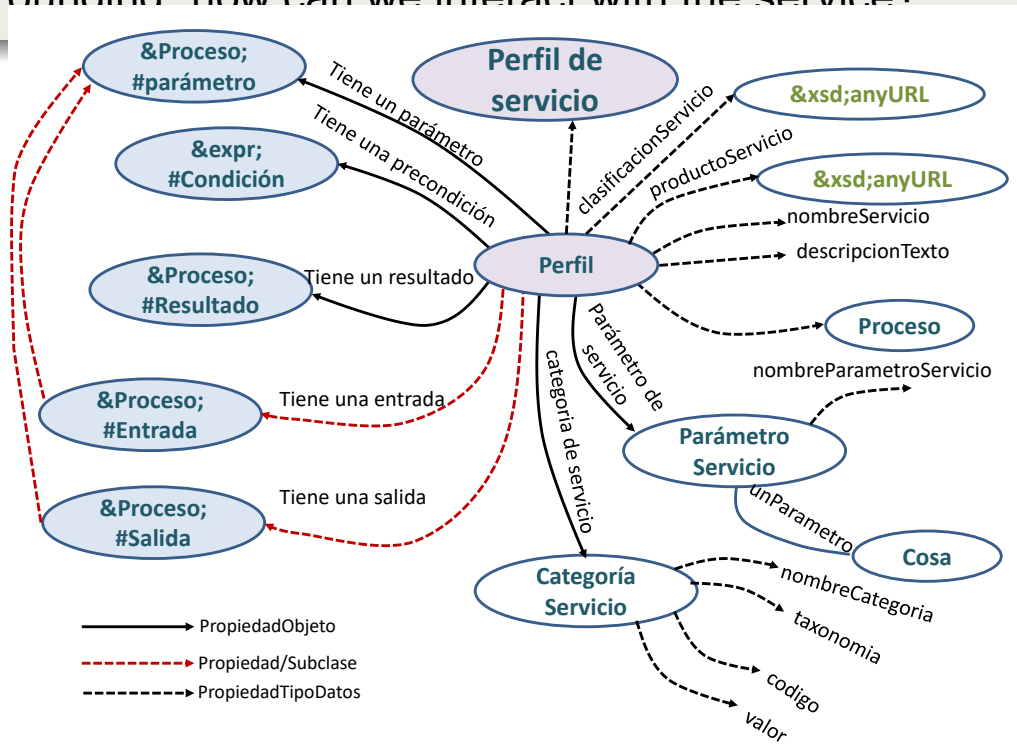
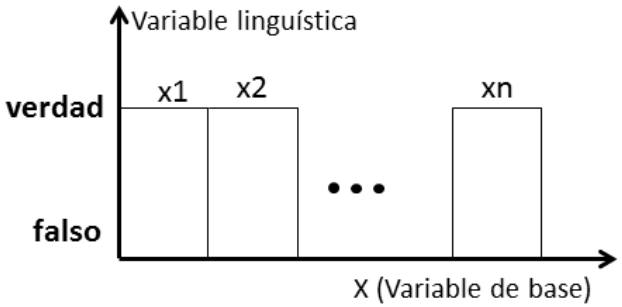
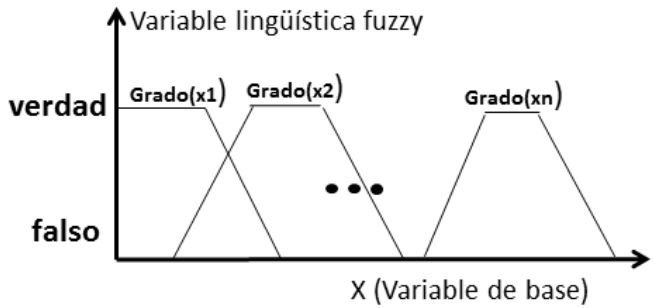


Figure: Selected classes and properties of one profile

Context-Aware services composition

Abstract representation of the *degree of membership* of a property to an interval –expressed as a base(X) variable– for the precise definition of the certainty of a property





Context-Aware services composition–II

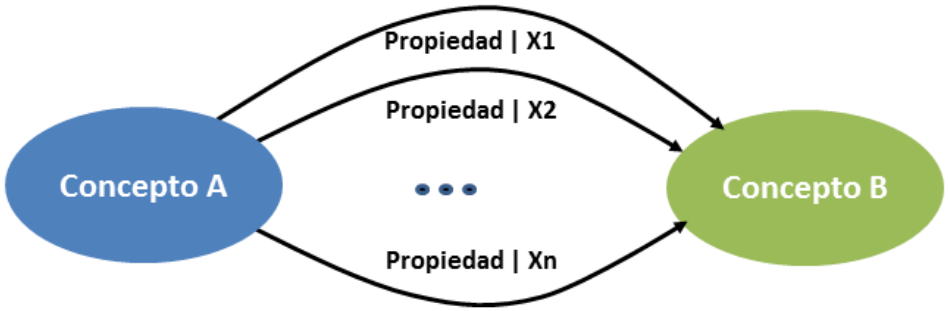


Figure: Abstract representation of the *degree of membership* of a property

Context-Aware services composition–III

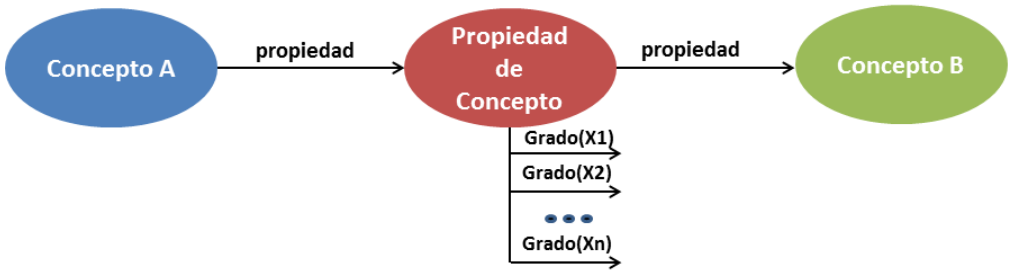


Figure: Degree of membership of a property to a certainty interval

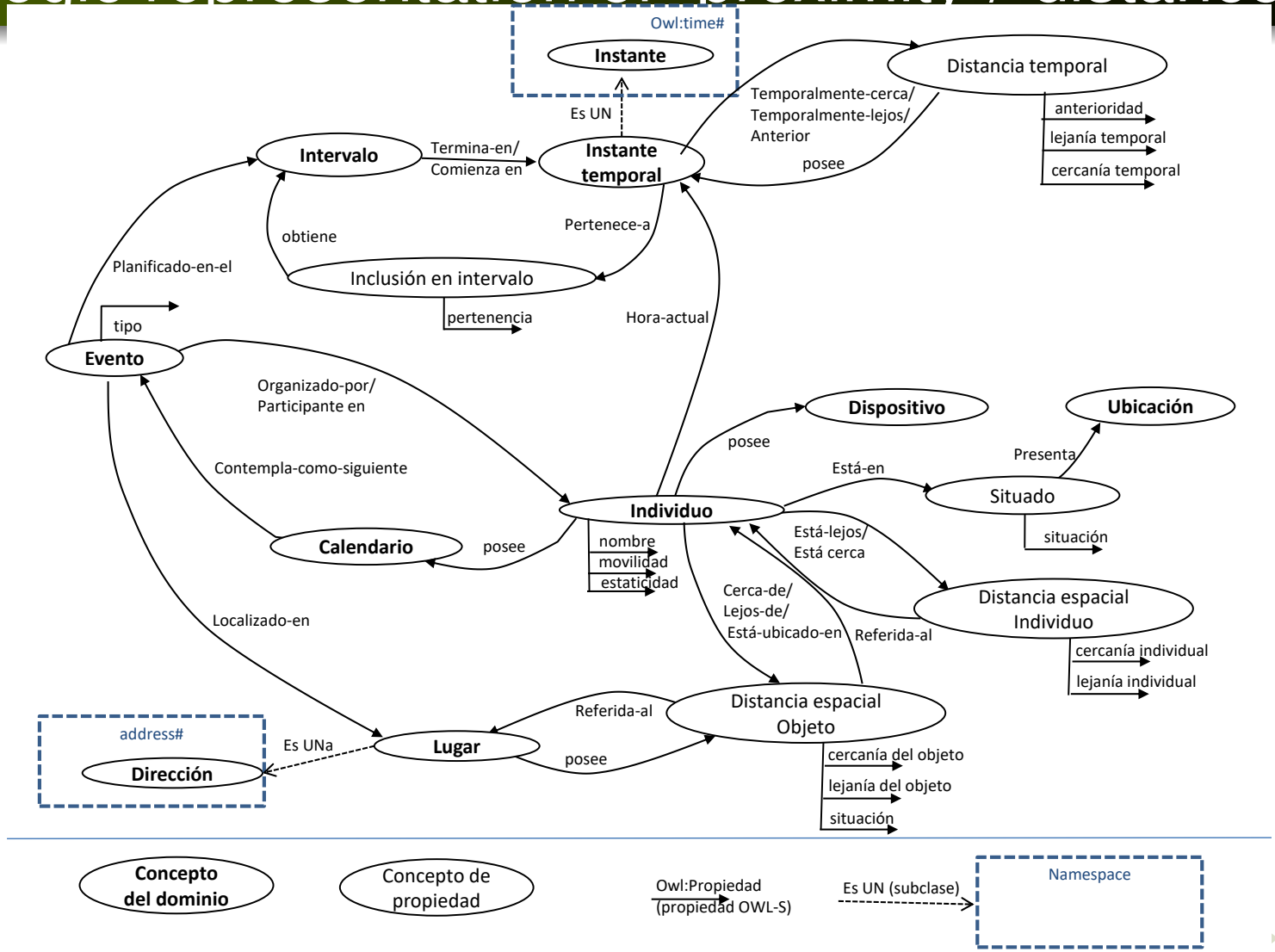
Representation pattern applicable to the properties related to the same base and concept pair

Context-Aware services composition–IV

Example of an ontologic representation of properties

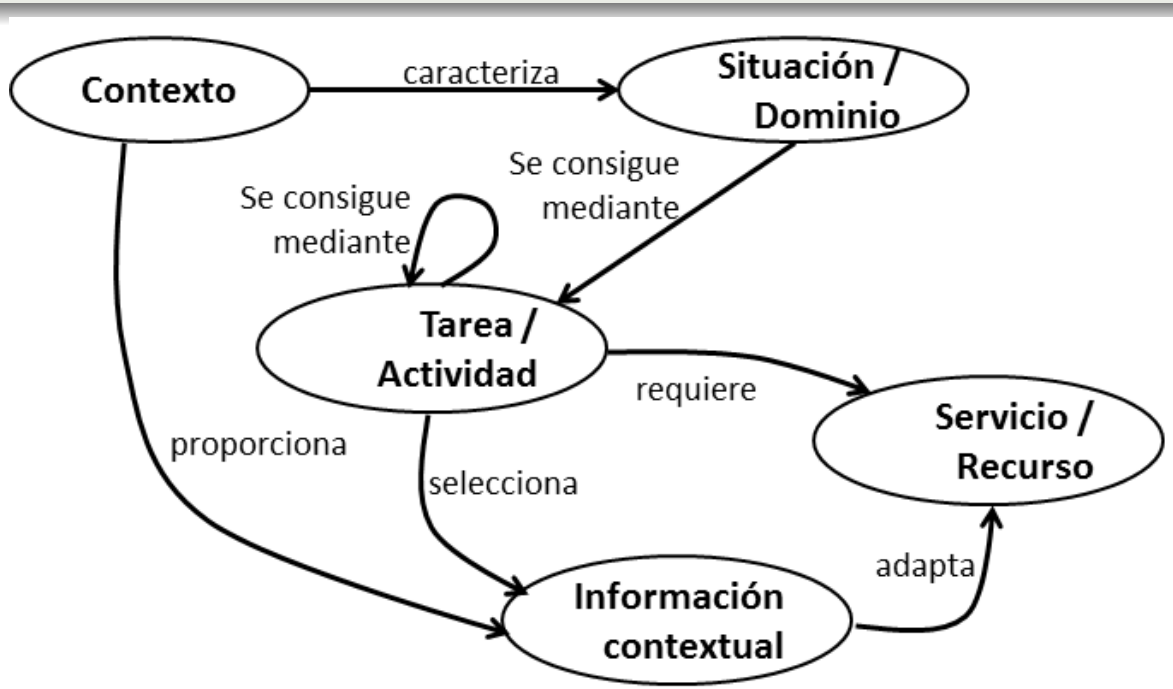
- Definition of “Proximity/Distance” of an individual to a place/time
- Property: Spatial/Temporal Distance
- Base variable: “distance” (over a range of values)
- Concepts: Individual, Place, Temporal instant, etc.
- The certainty in the fulfillment of properties will depend on the *membership degree* of the value of base-variable “distance” to an interval previously established

Ontologic representation of "proximity"/"distance"

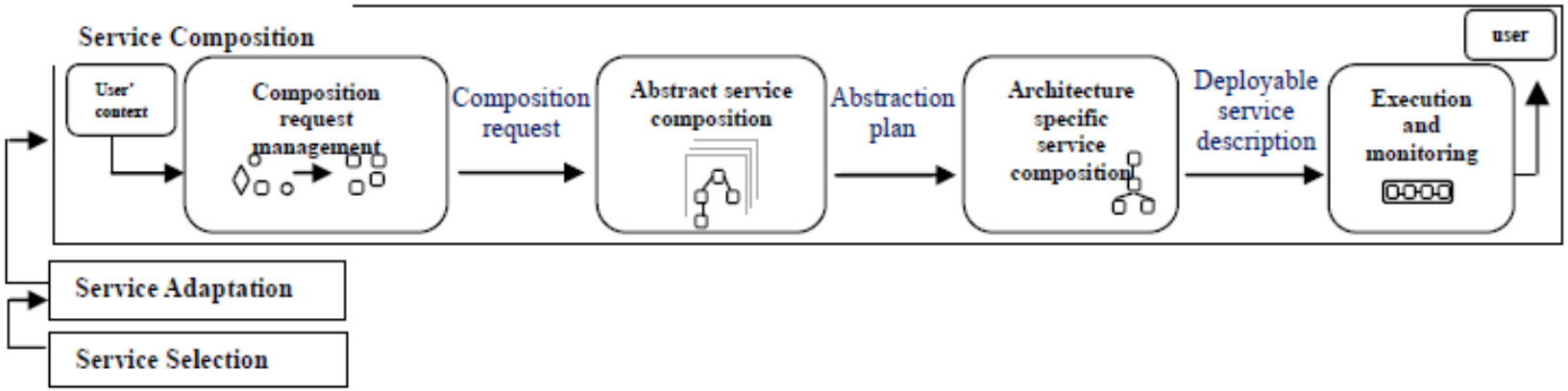


Rules associated to the ontologic model

Set of rules for inference of model's real situations with respect to the ontology for properties by using *base-variables* and the concept of membership to an interval of values



Services' composition fuzzy



Models for knowledge representation and ubiquitous systems

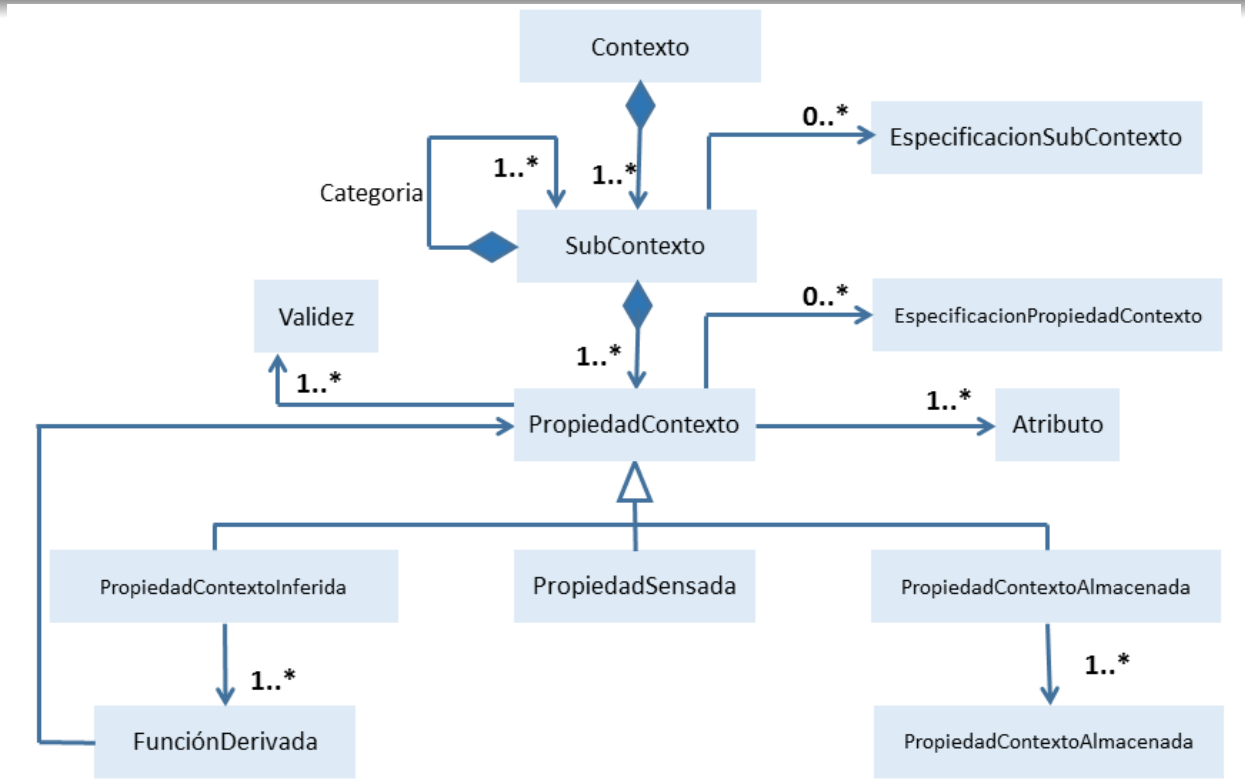


Figure: Metamodel of Ubiquitous Systems' Context

Knowledge Representation M. and Ubiquitous S.-II

Requirements approach	**Distributed composition**	**Partial verification**	**Richness and information quality**	**No complement. no complement**	**Formality level**	**Applicability to frameworks**
Key-value BD Models	−	−	−	−	−	+
Schemes or marking	+	++	−	−	+	++
Models graphs	−	-	+	−	+	+
OO Models	++	+	+	+	+	+
Logical Models	++	-	-	-	++	-
Ontological Models	++	++	+	+	++	+

Ontologies understood as a knowledge domain

- CONON [Wang et al., 2004]–CONtext ONtology
- CoDAMoS [Preuveneers et al., 2004] –Context–Driven Adaptation of MObile Services
- COBRA-ONT [Chen et al. , 2004]
- SOUPA [Chen et al., 2004] –Standard Ontology for Ubiquituous and Pervasive Applications–

Summary of ontologies that represent Ubiquitous-characteristic knowledge domains

Fundamental ideas

- We use ontologies to define words that can be accurately interpreted by machines
- An ontology is a way of representing knowledge within a specific context
- An ontology represents a data model linked to an information domain and the relationships between its objects
- Conceptualization is equivalent to ontologic representation:
  - abstract model of a concrete environment composed of concepts, properties and relationships
  - knowledge representation system that needs a language like OWL

Summary of ontologies that represent Ubiquitous-characteristic knowledge domains-II

[Maedche and Staab, 2004]

"an ontology is a logical theory formed by a vocabulary and a logical language, which in a domain of interest formalizes signs that describe things in the real world, allowing to make the correspondence between the signs and things in the most exact way possible"

- Ontologies allow for better semantic information processing
- They are a basic support for interoperability

Summary of ontologies that represent Ubiquitous-characteristic knowledge domains-III

Reusability issues

- Languages to represent ontologies suffer from the same problems as any XML-based language
- The reuse of onetologies is a hot topic of research today
- It is closely related to current research in the “Semantic Web”

Summary of ontologies that represent Ubiquitous-characteristic knowledge domains-IV

	CONON	CoDAMoS	Cobra-ONT	SOUPA
Dispositive	X	X	X	–
Context	–	X	X	–
Interface	–	–	–	–
Placement	X	X	X	X
Network	X	–	–	–
Role	–	X	X	–
Service	X	X	–	–
Time	–	X	X	X
User	X	X	X	X

Table: Summary of current ontologies for Ubiquitous systems

Current reasearch projects that use ontologies for ubiquitous systems context modeling

- Intelligent agents deployment + context changes adaptation in real-time [Machuca et al., 2005]
- SOAM [Vazquez et al., 2006] and intelligent objects creation that use the Sematic Web services
- Platform of services within a hybrid context [Ejigu et al., 2008]

Exercise

Formulation:

Draw an ontological diagram that will describe the semantic information regarding the user's context of a ubiquitous system, necessary for the correct composition of services for locating a restaurant suitable for a meeting with his friends. The user is subscribed to a mobile network provider, which maintains a portal with a channel of informacion+entertaintment (Infoaintment) for its users. *Infoaintment* offers a wide range of resources and services to its users: a point of information of interest, e-commerce and specialized search engines. The user uses his *smart-phone* strolls to the main square of his city and access Infoaintment, which gives him access to a restaurant recommendation service. The mentioned portal uses an adaptive infrastructure to customize the service by adapting it to the user's context who can request it. In that case, the service request has resulted in the construction of a service from 2 Atomic services adapted to the context of the user. The context is enhanced directly by subscribing to the portal of services *Infoaintment*

Solution to the exercise

To be completed

Figure: *composition of services for locating a restaurant suitable for a meeting*

/


Apache (2014).
Jini technology.
https://river.apache.org.


Atzori, L., Iera, A., and Morabito, F. (2010).
The Internet of Things: a survey.
Computer Networks, 54(15):2787–2805.


Baldauf, M., Dustdar, S., and Rosenberg, F. (2012).
A survey of context aware systems.
International Journal of Adhoc and Ubiquitous Computing, 2(4):263–267.


Banavar, G. etal. (2000).
Challenges: An application model for pervasive computing.
In *Proceedings of the Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 266–274.


Branca, G. and Atzori, L. (2012).
A survey of soa technologies in ngn network architectures.
IEEE Communications Surveys Tutorials, 14(3):644–661.


Geyik, S.C.,Szymanski, B., and Zerfos, P.
Robust Dynamic Service Composition in Sensor Networks.
In *IEEE Transactions on Services Computing.*


Guinard, D., Trifa, V., and etal. (2010).
Interacting with the soa-based internet of things: Discovery, query, selection and on–demand provisioning of web services.
IEEE Transactions on Services Computing, 3(3):223–235.


Kim, J., Lee, J., Kim, J., and Yun, J. (2014).
M2M service platforms: survey, issues, and enabling technologies.
IEEE Communications Surveys Tutorials, 16(1):61–76.


Lassila, O. (2005).
Using the semantic Web in mobile and ubiquitous computing, volume 188 of *Industrial Applications of Semantic Web–IFIP*.
Springer Verlag, Indianapolis,Ind.


Locke, S. (2006).
Context–aware pervarsive systems: architectures for a new breed of applications.
CRC Press.


Madkour, M., El Ghanami, D., Maach, A., and Hasbi, A.
Context–aware service composition: An approach based on fuzzy sets and service composition.
Journal on Information Science and Engineering, 29(1):1–16.


Oracle (2014).
JXTA technology.
https://java.net/projects/jxta.


Ou, S., Georgalas, M., Yang, K., and Sun, X. (2006).
"A model driven integration architecture for ontology–based context modelling and context–aware application development" In Model–driven architecture, foundations and applications.
LNCS. Springer–Verlag.


Perera, C., Zaslavsky, A., Christen, P., and etal. (2014).
Context–aware computing for the internet of things: A survey.
IEEE Communications Surveys Tutorials, 16(1):414–454.


Raz, D., Juhola, A., Serrat-Fernandez, J., and Galis, A. (2006).
Fast and efficient context–aware services.
John Wiley and Sons.


Stavropoulos, T., Vrakas, D., and Vlahavas, I. (2013).
A survey of service composition in ambient intelligence environments.
Artificial Intelligence Review, 40(3):247–270.


Vazquez, J., Sedano, I., and Lopez de Ipua, D. (2007).
A web–powered architecture for designing and deploying pervasive semantic devices.
Internatinal Journal of Web information systems, 2(3):212–214.


Weiser, M. (1993).
Some Computer Science issues in ubiquitous computing.
Communications of the ACM, 36(7):75–84.


Wang, X.H., Zhang, D.Q., Gu, T., and etal. (2004).
Ontology based context modeling and reasoning using OWL.
IEEE Annual Conference on Pervasive Computing and Communications, 18–22.